RADIANT + Think Tank Complete Documentation

Version: 4.18.3
Generated: $(date +%Y-%m-%d)

This document contains the complete documentation for the RADIANT platform and Think Tank AI. It is intended for comprehensive system analysis and review.

Table of Contents

Part 6: API & Reference

- API Reference
- Error Codes
- Troubleshooting

Document Generated: Sun Dec 28 16:28:04 PST 2025

PART 1: PLATFORM OVERVIEW

Executive Summary

RADIANT & Think Tank Executive Summary

Enterprise AI Platform Overview

Version 4.18.0 | December 2024

For executives, investors, and decision-makers

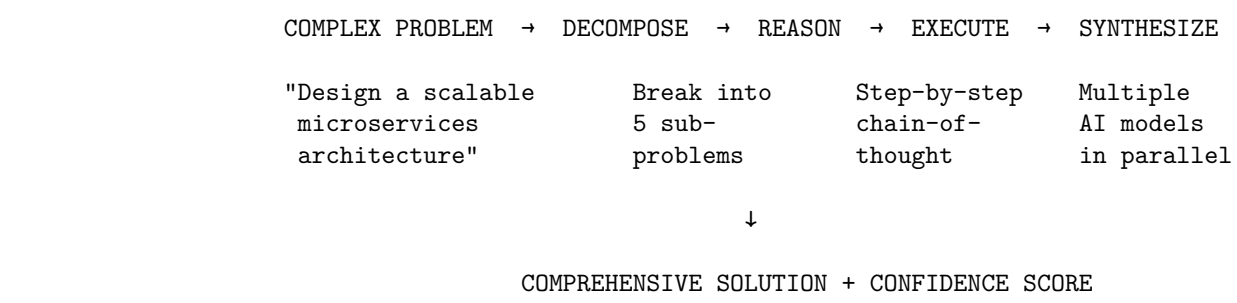
What is RADIANT?

RADIANT is an enterprise-grade, multi-tenant AI platform that provides organizations with unified access to 106+ AI models through a single API, with intelligent orchestration that coordinates multiple AI systems to deliver superior results.

```
                    ONE PLATFORM
                         ↓

106+ AI MODELS    •    49 ORCHESTRATION PATTERNS    •    9 MODES

OpenAI • Anthropic • Google • Meta • Mistral • DeepSeek • +10 more

                         ↓
```

What is Think Tank?

Think Tank is RADIANT's advanced problem-solving platform that decomposes complex problems into manageable steps, applies multi-AI reasoning, and synthesizes comprehensive solutions with confidence scoring.



Key Differentiators

1. AGI-Driven Model Selection

Unlike platforms that use a single AI model, RADIANT's AGI layer **automatically selects the optimal combination of models** based on task analysis:

What We Analyze	What We Select
Problem domain (coding, legal, medical…)	Best models for that domain
Task complexity	Number of models (2-5)
Reasoning requirements	Execution mode (thinking, fast, precise…)
Quality vs speed priority	Parallel execution strategy

Result: 50-300% better outcomes than single-model approaches.

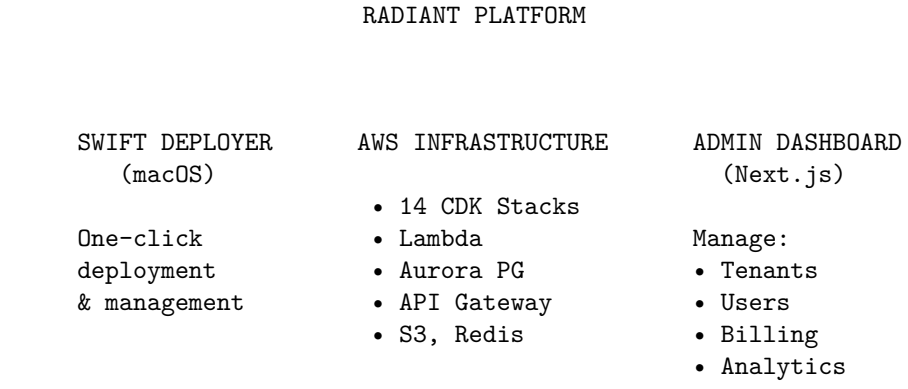
2. 49 Proven Orchestration Patterns

Research-backed workflows including: - **AI Debate** - Two AIs argue, judge decides - **Self-Refine** - Generate → Critique → Improve - **Chain-of-Verification** - Fact-check every claim - **Tree of Thoughts** - Explore multiple solution paths

3. Enterprise-Grade Security

Capability	Description
Multi-Tenant Isolation	PostgreSQL Row-Level Security
Compliance	SOC2, HIPAA-ready
Encryption	At-rest (AES-256) and in-transit (TLS 1.3)
Audit Logging	Complete activity trail

Platform Components



By the Numbers

Metric	Value
AI Models	106+ (50 external + 56 self-hosted)
AI Providers	15+ integrated
Orchestration Patterns	49 documented workflows
Execution Modes	9 specialized modes

Metric	Value
Database Migrations	66+ schema versions
CDK Stacks	14 infrastructure components

Use Cases

Enterprise AI Gateway

- Unified access to all major AI providers
- Centralized cost management and budgeting
- Consistent API regardless of backend model
- Automatic failover for reliability

Complex Problem Solving (Think Tank)

- Multi-step technical analysis
- Research synthesis with citations
- Architecture design with artifacts
- Decision support with confidence scores

Quality-Critical Applications

- Legal document analysis (precise mode)
- Medical information processing (HIPAA compliant)
- Financial analysis (multi-model verification)
- Code generation (AI debate + critique)

Cost Optimization

- Intelligent model routing (use cheaper models when appropriate)
- Budget alerts and limits
- Usage analytics by team/project
- Model performance vs cost analysis

Competitive Advantages

vs. Single-Model APIs	vs. Other Platforms
Multi-model orchestration	49 research-backed patterns
Built-in verification	AGI-driven model selection
Higher accuracy	9 execution modes
Reduced bias	Visual workflow editor

vs. Single-Model APIs	vs. Other Platforms
Confidence scoring	Think Tank problem solving

Technology Stack

Layer	Technology
Frontend	Next.js 14, TypeScript, Tailwind CSS, shadcn/ui
Backend	AWS Lambda (Node.js 20), API Gateway
Database	Aurora PostgreSQL (Serverless), DynamoDB, Redis
Infrastructure	AWS CDK (TypeScript), 14 stacks
Desktop	SwiftUI (macOS 13.0+, Swift 5.9+)
Security	Cognito, KMS, WAF, Row-Level Security

Deployment Model

RADIANT deploys to **your AWS account**:

```
YOUR AWS ACCOUNT

                    RADIANT INFRASTRUCTURE

  • Your data stays in your account
  • Your compliance requirements met
  • Your region/residency requirements
  • Full control over infrastructure

Deployed via Swift Deployer (macOS app) or CLI
```

Pricing Model

Tier	Target	Includes
Free	Developers	10K tokens/month, 3 models
Pro	Teams	1M tokens/month, all models, orchestration
Enterprise	Organizations	Unlimited, SLA, custom patterns, HIPAA

All tiers include: - Full API access - Admin dashboard - Basic analytics - Email support

---

Roadmap Highlights

Timeframe	Features
Q1 2026	Mobile SDK, more self-hosted models
Q2 2026	Fine-tuning pipeline, custom model hosting
Q3 2026	Multi-region deployment, advanced compliance
Q4 2026	Marketplace for custom patterns

Summary

RADIANT + Think Tank delivers:

1. **Unified AI Access** - One API for 106+ models across 15+ providers
2. **Intelligent Orchestration** - AGI selects optimal models and modes
3. **Superior Results** - 49 patterns achieve 50-300% better outcomes
4. **Enterprise Security** - Multi-tenant, SOC2, HIPAA-ready
5. **Cost Control** - Budgets, analytics, intelligent routing
6. **Problem Solving** - Think Tank for complex multi-step reasoning

---

RADIANT v4.18.0 + Think Tank v3.2.0

The enterprise platform for intelligent AI orchestration

Contact: info@radiant.ai | **Documentation:** docs.radiant.ai

---

RADIANT Architecture v4.17.0

Overview

RADIANT is a multi-tenant AWS SaaS platform for AI model access and orchestration. It consists of:

1. **Swift Deployer App** - macOS GUI for infrastructure deployment
2. **AWS Infrastructure** - CDK-based cloud infrastructure
3. **Admin Dashboard** - Next.js admin UI (Phase 3)

Technology Stack

Layer	Technology
Swift App	SwiftUI, macOS 13.0+, Swift 5.9+, GRDB
Infrastructure	AWS CDK (TypeScript), Aurora PostgreSQL, Lambda
Dashboard	Next.js 14, TypeScript, Tailwind CSS
AI Integration	106+ models, LiteLLM, SageMaker

Monorepo Structure

```
radiant/
  packages/
    shared/              # @radiant/shared - Types & constants
    infrastructure/      # @radiant/infrastructure - CDK stacks
  apps/
    swift-deployer/      # RadiantDeployer macOS app
  functions/              # Lambda functions (Phase 2)
  migrations/             # Database migrations (Phase 2)
  docs/                   # Specifications
```

Phase 1 Architecture

@radiant/shared Package

Single source of truth for all types and constants:

```
// Types
- app.types.ts        // ManagedApp, DeploymentStatus, etc.
- environment.types.ts // Environment, TierConfig, etc.
- ai.types.ts         // AIProvider, AIModel, etc.
- admin.types.ts      // Administrator, Permissions
- billing.types.ts    // UsageEvent, Invoice, etc.
- compliance.types.ts // PHI, AuditLog, etc.
```

```
// Constants
- version.ts          // RADIANT_VERSION = "4.17.0"
- tiers.ts            // Tier 1-5 configurations
- regions.ts          // AWS region configs
- providers.ts        // AI provider definitions
```

Swift Deployer App

```
RadiantDeployer/
   RadiantDeployerApp.swift  # @main entry point
   AppState.swift            # Global state management
   Models/
       ManagedApp.swift      # App configuration model
       Credentials.swift     # AWS credentials model
       Deployment.swift      # Deployment state/progress
   Services/
       CredentialService.swift  # Keychain credential storage
       CDKService.swift         # CDK command execution
       AWSService.swift         # AWS API interactions
   Views/
        MainView.swift        # NavigationSplitView container
        AppsView.swift        # Application grid
        DeployView.swift      # Deployment wizard
        ProvidersView.swift   # AI provider list
        ModelsView.swift      # AI model catalog
        SettingsView.swift    # Preferences
```

CDK Infrastructure Stacks

```
packages/infrastructure/
   bin/radiant.ts            # CDK app entry point
   lib/stacks/
       foundation-stack.ts   # KMS, SSM parameters
       networking-stack.ts   # VPC, subnets, endpoints
       security-stack.ts     # Security groups, WAF
```

Infrastructure Tiers

Tier	Name	VPC CIDR	AZs	NAT	Aurora ACU	Features
1	SEED	/20	2	1	0.5-2	Dev only
2	STARTUP	/18	2	1	1-8	WAF, GuardDuty
3	GROWTH	/17	3	2	2-16	Self-hosted models, HIPAA
4	SCALE	/16	3	3	4-64	Multi-region, Global DB
5	ENTERPRISE	/14	3	3	8-128	Full enterprise

Deployment Flow

```
Swift Deployer
   (macOS)

         1. Configure credentials
         2. Select app/tier/env
         3. Initiate deployment


  CDK Deploy
 (via Process)

         4. Bootstrap AWS
         5. Synth stacks
         6. Deploy in order


 AWS Account
 (VPC, RDS,
  Lambda, etc)
```

Phase 1 Deliverables

- ☒ Monorepo structure (pnpm workspace)
- ☒ @radiant/shared package with all types
- ☒ @radiant/infrastructure base CDK stacks
- ☒ RadiantDeployer Swift app with full UI
- ☒ Credential management via Keychain
- ☒ CDK service for deployment execution
- ☒ Documentation

Future Phases

Phase	Sections	Description
2	3-7	AI stacks, Lambdas, Database
3	8-9	Admin Dashboard, Deployment Guide
4	10-17	Visual AI, Brain, Analytics
5	18-28	Think Tank, Collaboration
6	29-35	Registry, Time Machine
7	36-39	Neural Engine, Workflows
8	40-42	Isolation, i18n, Config
9	43-46	Billing System

PART 2: ADMINISTRATION GUIDES

RADIANT Admin Guide

RADIANT Platform - Administrator Guide

Complete guide for managing the RADIANT AI Platform via the Admin Dashboard

Version: 4.18.1 | Last Updated: December 2024

Table of Contents

1. Introduction

1.1 What is RADIANT?

RADIANT is a multi-tenant AWS SaaS platform providing unified access to 106+ AI models through:

- **50 External Provider Models**: OpenAI, Anthropic, Google, xAI, DeepSeek, and more
- **56 Self-Hosted Models**: Running on AWS SageMaker for cost control and privacy
- **Intelligent Routing**: Brain router for optimal model selection
- **Neural Engine**: Personalization learning from user interactions

1.2 Administrator Roles

Role	Permissions	Use Case
Super Admin	Full access to all features	Platform owner
Admin	Tenant management, billing, models	Operations team
Operator	Read access, limited actions	Support team
Auditor	Read-only access to logs	Compliance team

Role Details **Super Admin** - The highest privilege level with unrestricted access: - Create and delete tenants - Manage all administrators - Access all billing and financial data - Modify system-wide configuration - Approve production database migrations - Impersonate any tenant for debugging - Access compliance and audit reports - Typically limited to 1-3 people (CTO, lead engineer)

Admin - Day-to-day operations management: - Create and modify tenants (cannot delete) - Manage users within tenants - Configure AI models and providers - View billing data (cannot modify pricing) - Monitor system health - Cannot access other admin accounts - Typically assigned to operations team members

Operator - Limited support and monitoring: - View tenant information (read-only) - View user issues and support tickets - Monitor system health dashboards - Cannot modify any configuration - Cannot access billing or sensitive data - Typically assigned to support staff

Auditor - Compliance and security review: - Full read access to audit logs - Access to compliance reports - Cannot modify anything - Cannot view sensitive data (API keys, passwords) - Access is logged for compliance - Typically assigned to compliance officers or external auditors

1.3 Key Concepts

Concept	Description
Tenant	Organization with isolated data
User	End-user within a tenant
Subscription	Billing tier (1-7)
Credits	Currency for AI usage
API Key	Authentication for API access
App	Consumer application (Think Tank, etc.)

Tenant Architecture Explained A **Tenant** represents a complete organization using RADIANT. Each tenant has:

- **Complete Data Isolation**: All data is stored with tenant IDs and protected by PostgreSQL Row-Level Security (RLS). One tenant can never access another tenant's data, even if there's a bug in application code.
- **Separate Billing**: Each tenant has its own subscription, credit balance, and usage tracking. Costs are attributed to the correct tenant automatically.
- **Custom Configuration**: Tenants can customize model access, rate limits, and feature flags without affecting other tenants.
- **User Management**: Each tenant manages their own users, roles, and permissions independently.

User vs Administrator **Users** are end-users who interact with RADIANT-powered applications like Think Tank. They: - Sign up and log in via Cognito - Use AI models through the API or applications - Have credits deducted for usage - Cannot access the Admin Dashboard

Administrators manage the RADIANT platform itself. They: - Access the Admin Dashboard - Manage tenants, users, and billing - Configure AI models and providers - Have no credits (administrative access is separate)

Credit System Explained Credits are RADIANT's universal currency for AI usage:

- **1 credit = $0.01 USD** (configurable per deployment)
- Different models cost different amounts based on their API pricing
- Credits are deducted in real-time as requests complete
- Tenants can purchase credits or receive them through subscriptions
- Credits can be tracked, audited, and reported on

Example Credit Costs: | Model | Cost per 1K tokens | |——-|——————-| | GPT-4o | 5 credits input, 15 credits output | | GPT-4o-mini | 0.5 credits input, 1.5 credits output | | Claude 3.5 Sonnet | 3 credits input, 15 credits output | | Self-hosted Llama | 0.2 credits (all) |

API Key Types RADIANT supports multiple API key types:

- **User API Keys**: Tied to a specific user, inherit user's permissions
- **Service API Keys**: For server-to-server communication, not tied to a user
- **Admin API Keys**: For administrative operations, require elevated permissions
- **Scoped Keys**: Limited to specific models, endpoints, or rate limits

---

2. Accessing the Admin Dashboard

2.1 URL and Login

1. Navigate to: `https://admin.your-domain.com`
2. Enter your email address
3. Enter your password
4. Complete MFA verification (required)

2.2 First Login

On first login:

1. You'll receive a temporary password via email
2. Enter the temporary password
3. Set a new password (12+ characters, mixed case, numbers, symbols)
4. Set up MFA using an authenticator app
5. You'll be redirected to the dashboard

2.3 Session Management

Setting	Value
Session Duration	8 hours
Idle Timeout	30 minutes
Concurrent Sessions	3 maximum
Remember Device	30 days

2.4 Password Requirements

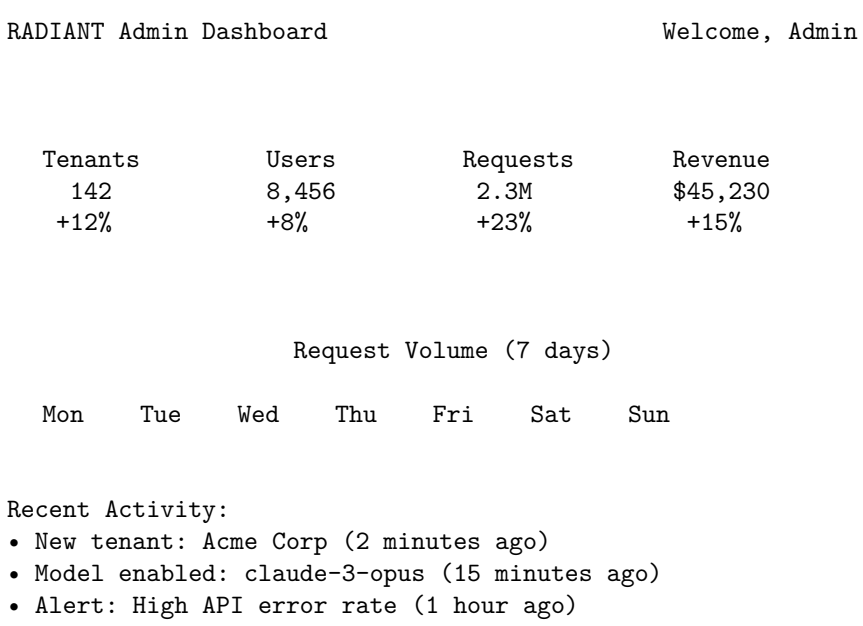
- Minimum 12 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one number
- At least one special character
- Cannot reuse last 10 passwords

---

3. Dashboard Overview

3.1 Main Dashboard

The dashboard displays key metrics at a glance:



3.2 Navigation Menu

Section	Description
Dashboard	Overview and metrics
Tenants	Tenant management
Users	User management
Models	AI model configuration
Providers	Provider management
Billing	Subscriptions and credits
Storage	Storage usage
Orchestration	Neural engine settings
Localization	Translation management
Configuration	System settings
Security	Security monitoring
Compliance	Compliance reports
Experiments	A/B testing
Cost	Cost analytics

Section	Description
Audit	Audit logs
Migrations	Database migrations
Notifications	System alerts
Settings	Personal settings

---

4. Tenant Management

4.1 Viewing Tenants

Navigate to **Tenants** to see all organizations:

Column	Description
Name	Organization name
Plan	Subscription tier
Users	User count
Status	Active/Suspended/Trial
Created	Creation date
Last Active	Last API call

4.2 Creating a Tenant

1. Click **"+ New Tenant"**
2. Fill in required fields:
   - **Name**: Organization name
   - **Slug**: URL-friendly identifier
   - **Plan**: Initial subscription tier
   - **Admin Email**: Primary admin email
3. Configure optional settings:
   - Custom domain
   - Branding settings
   - Feature flags
4. Click **"Create Tenant"**

4.3 Tenant Details

View comprehensive tenant information:

```
Tenant: Acme Corporation

Overview          Users          Billing          Settings
```

```
Tenant ID:        tn_abc123xyz
Status:            Active
Plan:             Professional (Tier 4)
Created:          2024-01-15
Last Active:      2 minutes ago

Usage This Month:
   API Requests:     145,234
   Tokens Used:      12.5M
   Storage:          2.3 GB
   Credits Used:     $1,234.56

[Edit]  [Suspend]  [Delete]  [Impersonate]
```

4.4 Tenant Actions

Action	Description	Permission
Edit	Modify tenant settings	Admin
Suspend	Temporarily disable	Admin
Delete	Permanently remove	Super Admin
Impersonate	Login as tenant admin	Super Admin
Export	Export tenant data	Admin

4.5 Data Isolation

Each tenant has complete data isolation:

- Separate database rows with RLS
- Unique API keys
- Isolated storage buckets
- Independent usage tracking

---

5. User & Administrator Management

5.1 Administrator Roles

Role	Dashboard Access	API Access	Billing	Audit
Super Admin	Full	Full	Full	Full
Admin	Full	Full	Read	Read

Role	Dashboard Access	API Access	Billing	Audit
Operator	Read	Read	None	Read
Auditor	Logs only	None	None	Full

5.2 Managing Administrators

Navigate to **Administrators** to:

1. **Invite New Admin**:
   - Click **"+ Invite Administrator"**
   - Enter email address
   - Select role
   - Click **"Send Invitation"**
2. **Modify Admin**:
   - Click on administrator row
   - Edit role or permissions
   - Click **"Save Changes"**
3. **Remove Admin**:
   - Click **"Remove"** button
   - Confirm removal
   - Admin's sessions are invalidated immediately

5.3 Viewing Tenant Users

Navigate to **Tenants** → [**Tenant**] → **Users** to see:

Field	Description
Email	User email
Name	Display name
Role	Tenant role
Status	Active/Invited/Disabled
Last Login	Last authentication
API Keys	Number of active keys

5.4 User Actions

Action	Description
Reset Password	Send password reset email
Disable	Prevent login
Enable	Restore access
Delete	Remove user data
View Sessions	See active sessions

6. AI Model Configuration

6.1 Model Registry

Navigate to **Models** to see all available models:

```
AI Models                                              106 Total


Filter: [All ]  Category: [All ]  Status: [Enabled ]

Model               Provider    Category   Tier   Status

gpt-4o              OpenAI      Chat       1        Enabled
gpt-4o-mini         OpenAI      Chat       1        Enabled
claude-3-opus       Anthropic   Chat       2        Enabled
claude-3-sonnet     Anthropic   Chat       1        Enabled
gemini-pro          Google      Chat       1        Enabled
llama-3.1-70b       Self-Host   Chat       3        Enabled
whisper-large       Self-Host   Audio      3        Disabled

[+ Add Model]  [Import Models]  [Export Config]
```

6.2 Model Categories

Category	Description	Example Models
Chat/LLM	Text generation	GPT-4o, Claude 3, Gemini
Embedding	Vector embeddings	text-embedding-3-large
Vision	Image understanding	GPT-4V, Claude Vision
Audio	Speech-to-text	Whisper, Deepgram
Image	Image generation	DALL-E 3, Stable Diffusion
Code	Code generation	Codestral, DeepSeek Coder
Scientific	Research models	BioGPT, ChemLLM

Category Details **Chat/LLM (Large Language Models)**: The core of RADIANT. These models handle conversational AI, content generation, summarization, and general-purpose text tasks. They're the most commonly used and include flagship models from OpenAI, Anthropic, Google, and open-source alternatives.

Embedding Models: Convert text into numerical vectors for semantic search, similarity matching, and retrieval-augmented generation (RAG). Essential for

building knowledge bases and search functionality. Vectors are typically 1536-3072 dimensions.

Vision Models: Analyze images, extract text (OCR), describe visual content, and answer questions about images. Increasingly important for document processing, accessibility, and multimodal applications.

Audio Models: Transcribe speech to text, translate audio, and identify speakers. Whisper is the most popular, offering excellent accuracy across 99 languages. Used for meeting transcription, accessibility, and voice interfaces.

Image Generation: Create images from text descriptions. DALL-E 3 offers the best prompt following, while Stable Diffusion provides more customization options. Consider content policies when enabling these.

Code Models: Specialized for programming tasks including code generation, explanation, debugging, and refactoring. Some are fine-tuned on specific languages or frameworks.

Scientific Models: Domain-specific models trained on scientific literature. Useful for research applications but require careful evaluation for accuracy.

6.3 Model Configuration

Click on a model to configure:

Setting	Description
Enabled	Available for use
Min Tier	Minimum subscription tier
Rate Limits	Requests per minute
Max Tokens	Maximum context/output
Temperature Range	Allowed temperature values
Price Override	Custom pricing

Configuration Settings Explained **Enabled**: When disabled, the model is hidden from users and API requests return "model not found". Use this to temporarily remove models during maintenance or to restrict access to specific models.

Min Tier: Sets the minimum subscription tier required to access this model. For example, setting GPT-4 to Tier 2 means Free tier users cannot use it. This helps control costs and create upgrade incentives.

Rate Limits: Controls requests per minute per user for this model. Prevents abuse and ensures fair access. Set based on the provider's rate limits and your capacity: - Conservative: 10-20 requests/minute - Standard: 50-100 requests/minute
- High: 200+ requests/minute (requires provider rate limit increases)

Max Tokens: Limits context window and output length. Useful for controlling costs since longer contexts cost more. Set based on use case: - Short tasks (Q&A): 4,096 tokens - Medium tasks (writing): 16,384 tokens - Long tasks (analysis): 32,768+ tokens

Temperature Range: Restricts the temperature parameter users can set. Temperature controls randomness: - 0.0: Deterministic, consistent outputs - 0.7: Balanced creativity and consistency - 1.0+: More creative, less predictable

Restricting range (e.g., 0.0-1.0) prevents users from setting extreme values that produce poor results.

Price Override: Allows custom pricing different from the default. Useful for: - Offering discounts on specific models - Increasing prices for premium models - Matching competitor pricing - A/B testing pricing strategies

6.4 Self-Hosted Models

For Tier 3+ deployments:

1. Navigate to **Models → Self-Hosted**
2. Click **"+ Add Self-Hosted Model"**
3. Configure:
   - **Model ID**: Unique identifier
   - **SageMaker Endpoint**: Endpoint name
   - **Instance Type**: ml.g5.xlarge, etc.
   - **Auto-Scaling**: Min/max instances
4. Deploy model to SageMaker

6.5 Thermal States (Self-Hosted)

State	Description	Response Time
HOT	Always running	<100ms
WARM	Scaled down	<5s
COLD	Stopped	30-60s
OFF	Disabled	N/A

---

7. Provider Management

7.1 External Providers

Navigate to **Providers** to manage API integrations:

Provider	Models	Status	Health
OpenAI	12	Configured	99.9%
Anthropic	6	Configured	99.8%
Google AI	8	Configured	99.7%
xAI	2	Configured	99.5%
DeepSeek	4	Not configured	-

7.2 Adding Provider Credentials

1. Click on provider name
2. Click **"Configure"**
3. Enter API credentials:
   - **API Key**: Provider API key
   - **Organization ID**: (if applicable)
   - **Base URL**: (for custom endpoints)
4. Click **"Test Connection"**
5. Click **"Save"**

7.3 Provider Health Monitoring

View real-time provider health:

```
Provider Health: OpenAI


Status:             Healthy
Uptime (30d):     99.94%
Avg Latency:      245ms
P95 Latency:      520ms
Error Rate:       0.02%

Last 24 Hours:

12am        6am        12pm        6pm        12am
```

7.4 Fallback Configuration

Configure provider fallbacks:

1. Navigate to **Providers → Fallbacks**
2. Set priority order for each model category
3. Configure automatic failover rules
4. Set retry policies

8. Billing & Subscriptions

8.1 Subscription Tiers

Tier	Name	Monthly	Features
1	Free	$0	Basic models, 1K requests
2	Starter	$29	More models, 10K requests
3	Professional	$99	All external models, 100K requests
4	Business	$299	Priority support, 500K requests
5	Enterprise	$999	Self-hosted, unlimited
6	Enterprise+	Custom	Custom SLAs, dedicated support
7	Ultimate	Custom	On-premise options

8.2 Credit System

Credits are the universal currency for AI usage:

Model Type	Cost per 1M Tokens
GPT-4o	500 credits
GPT-4o-mini	50 credits
Claude 3 Opus	600 credits
Claude 3 Sonnet	150 credits
Self-hosted	20 credits

8.3 Managing Subscriptions

Navigate to **Billing → Subscriptions**:

1. View current subscription
2. Upgrade/downgrade tier
3. Add credit packages
4. View invoices
5. Update payment method

8.4 Usage Reports

Generate usage reports:

1. Navigate to **Billing → Reports**
2. Select date range
3. Choose grouping (by tenant/model/user)
4. Export as CSV/PDF

8.5 Billing Alerts

Configure alerts for:

- Credit balance low
- Usage spike
- Approaching quota
- Failed payments

---

9. Storage Management

9.1 Storage Overview

Navigate to **Storage** to monitor:

```
Storage Overview


Total Used:       234.5 GB of 500 GB (47%)


By Type:
   Documents:     120.3 GB (51%)
   Images:        45.2 GB (19%)
   Audio:         38.7 GB (17%)
   Video:         22.1 GB (9%)
   Other:         8.2 GB (4%)

Top Tenants:
1. Acme Corp       45.2 GB
2. TechStart       32.1 GB
3. DataCo          28.4 GB
```

9.2 Storage Tiers

Tier	Included	Additional
Free	1 GB	N/A
Starter	10 GB	$0.10/GB
Professional	100 GB	$0.08/GB
Business	500 GB	$0.05/GB
Enterprise	2 TB	$0.03/GB

9.3 File Management

Manage uploaded files:

- View file metadata
- Download files
- Delete files
- Set retention policies

---

10. Orchestration & Neural Engine

10.1 Brain Router

The Brain Router automatically selects optimal models:

Factor	Weight	Description
Cost	30%	Price optimization
Quality	30%	Output quality
Speed	20%	Response latency
Availability	20%	Provider health

10.2 Neural Patterns

Configure orchestration patterns:

Pattern	Description	Use Case
Single	One model	Simple requests
Fallback	Primary + backup	High availability
Parallel	Multiple simultaneous	Consensus
Chain	Sequential models	Complex tasks

10.3 Workflow Templates

Create reusable workflows:

1. Navigate to **Orchestration → Workflows**
2. Click **"+ New Workflow"**
3. Define steps and conditions
4. Set triggers and parameters
5. Save and activate

---

11. Pre-Prompt Learning

Location: Admin Dashboard → Orchestration → Pre-Prompts

The Pre-Prompt Learning system tracks and learns from the effectiveness of pre-prompts (system prompts) used by the AGI Brain. It uses **attribution analysis** to determine what factor actually caused issues - not just the pre-prompt.

11.1 Overview Dashboard

The dashboard shows key metrics:

Metric	Description
Templates	Active/total pre-prompt templates
Total Uses	Pre-prompt instances used
Avg Rating	Average user rating (1-5)
Thumbs Up Rate	Percentage of positive feedback
Exploration Rate	Rate of trying non-optimal templates to learn

11.2 Attribution Analysis

When users report issues, the system analyzes which factor was responsible:

Factor	When Blamed
Pre-prompt	System instructions wrong
Model	Model selection inappropriate
Mode	Orchestration mode wrong
Workflow	Pattern did not fit task
Domain	Domain detection incorrect
Other	External factors

The attribution pie chart shows the distribution of blame across factors.

11.3 Template Management

Navigate to the **Templates** tab to:

- View all pre-prompt templates
- See usage statistics and success rates
- Check which orchestration modes each template supports
- View average feedback scores

11.4 Adjusting Weights

Click the **sliders icon** on any template to adjust weights:

Weight	Default	Description
Base Effectiveness	0.5	Starting score before bonuses
Domain Weight	0.2	Bonus for matching domain
Mode Weight	0.2	Bonus for matching mode
Model Weight	0.2	Bonus for compatible model
Complexity Weight	0.15	Bonus for complexity match
Task Type Weight	0.15	Bonus for task type match
Feedback Weight	0.1	Historical feedback influence

Score Formula:

```
Final Score = Base + (Domain × DomainWeight) + (Mode × ModeWeight) +
              (Model × ModelWeight) + (Complexity × ComplexityWeight) +
              (TaskType × TaskTypeWeight) + FeedbackAdjustment
```

11.5 Learning Configuration

Configure learning behavior:

- **Learning Enabled**: Toggle on/off
- **Exploration Rate**: Percentage of requests using non-optimal templates to gather data (default: 10%)
- **Exploration Decay**: How quickly exploration decreases (default: 0.99)
- **Minimum Exploration**: Floor for exploration rate (default: 1%)

11.6 Recent Feedback

The **Feedback** tab shows recent user feedback with:

- Rating (1-5 stars)
- Attribution label (what got blamed)
- User comments
- Timestamp

See Pre-Prompt Learning System Documentation for full details.

---

12. Localization

12.1 Translation Management

Navigate to **Localization** to manage:

- Supported languages

- Translation strings
- AI translation settings

11.2 Supported Languages

Language	Code	Status
English	en	Default
Spanish	es	Enabled
French	fr	Enabled
German	de	Enabled
Japanese	ja	Enabled
Chinese	zh	Enabled

11.3 AI Translation

Enable AI-powered translation:

1. Navigate to **Localization → Settings**
2. Enable **"AI Translation"**
3. Select translation model
4. Configure quality settings

---

12. Configuration Management

12.1 System Configuration

Navigate to **Configuration** to manage:

Category	Settings
General	Platform name, domain, timezone
Email	SMTP settings, templates
Security	Password policy, MFA settings
API	Rate limits, CORS settings
Features	Feature flags

12.2 Tenant Overrides

Allow tenant-specific configuration:

1. Navigate to **Configuration → Tenant Overrides**
2. Select tenant
3. Override specific settings
4. Save changes

12.3 SSM Parameters

System configuration is stored in AWS SSM:

Parameter	Description
`/radiant/prod/database/url`	Database connection
`/radiant/prod/api/rate-limit`	API rate limits
`/radiant/prod/features/*`	Feature flags

---

13. AI Ethics & Standards

Location: Admin Dashboard → Ethics

The Ethics module provides transparency into the ethical principles guiding AI behavior and their source standards.

13.1 Standards Tab

View all industry AI ethics frameworks that inform RADIANT's ethical principles:

Standard	Organization	Type	Required
NIST AI RMF 1.0	National Institute of Standards and Technology	Government	
ISO/IEC 42001:2023	International Organization for Standardization	ISO	
EU AI Act	European Union	Government	
IEEE 7000-2021	IEEE	Industry	
OECD AI Principles	OECD	Government	
UNESCO AI Ethics	UNESCO	Government	
ISO/IEC 23894:2023	ISO	ISO	
ISO/IEC 38507:2022	ISO	ISO	

Each standard shows: - **Full Name**: Complete standard title with version - **Organization**: Issuing body - **Type**: Government, ISO, Industry, Academic,

Religious - **Required Badge**: Mandatory for compliance - **Description**: Summary of the standard's purpose - **Publication Date**: When the standard was issued - **External Link**: Direct link to official standard

13.2 Principles Tab

View ethical principles with their standard sources:

Principle	Category	Source Standards
Love Others	Love	NIST GOVERN 1.2, ISO 42001 Clause 5.2, Matthew 22:39
Golden Rule	Love	NIST MAP 1.1, EU AI Act Article 9, Matthew 7:12
Speak Truth	Truth	NIST GOVERN 4.1, ISO 42001 Clause 7.4, EU AI Act Article 13, John 8:32
Show Mercy	Mercy	NIST MEASURE 2.6, EU AI Act Article 14, Matthew 5:7
Serve Humbly	Service	NIST GOVERN 1.1, ISO 42001 Clause 5.1, Mark 10:45
Avoid Judgment	Mercy	NIST MAP 2.3, EU AI Act Article 10, Matthew 7:1
Care for Vulnerable	Service	NIST MAP 1.5, EU AI Act Article 7, ISO 42001 Clause 8.4, Matthew 25:40

Each principle displays: - **Teaching**: The principle text - **Category Badge**: love, mercy, truth, service, humility, peace, forgiveness - **Derived from / Aligned with**: Standard badges with section references

13.3 Violations Tab

Monitor ethical violations in AI responses:

- **Action**: What was evaluated
- **Score**: Ethical score percentage
- **Concerns**: Specific violations detected
- **Guidance**: Recommended corrections
- **Reasoning**: Explanation of the evaluation

13.4 Statistics

Summary cards showing: - **Total Evaluations**: All ethical checks performed - **Approved**: Passed evaluations - **Violations**: Failed evaluations - **Average Score**: Mean ethical score - **Today**: Evaluations in the last 24 hours

13.5 Standard Alignment Levels

Principles map to standards with different alignment levels:

Level	Meaning
derived	Principle was directly derived from this standard
aligned	Principle aligns with this standard's requirements
supports	Principle supports this standard's goals
extends	Principle extends beyond this standard

See AI Ethics Standards Documentation for full details.

---

14. Provider Rejection Handling

Location: Think Tank → Notifications (Bell icon)

When AI providers reject prompts based on their policies (not RADIANT's), the system automatically attempts fallback to alternative models.

14.1 How It Works

```
User Prompt → Model Rejects → Check RADIANT Ethics
     Our ethics block → Reject with explanation
     Our ethics pass → Try fallback models (up to 3 attempts)
           Fallback succeeds → Return response (user notified)
           All fail → Reject with full explanation
```

14.2 Rejection Types

Type	Description	Auto-Fallback
content_policy	Provider's content policy violation	Yes
safety_filter	Safety/moderation filter triggered	Yes
provider_ethics	Provider's ethical guidelines differ	Yes
capability_mismatch	Model can't handle request type	Yes
context_length	Prompt too long for model	Yes
moderation	Pre-flight moderation blocked	Yes
rate_limit	Rate limiting	Retry

14.3 User Notifications

Users see rejection notifications via the bell icon in Think Tank:

- **Unread count badge** - Number of unread notifications

- **Notification panel** - Slides out showing all rejections
- **Suggested actions** - Rephrase, simplify, contact admin
- **Resolution status** - Whether fallback succeeded

14.4 Fallback Model Selection

Models are selected for fallback based on:

1. **Rejection rate** - Models with lowest historical rejection rates preferred
2. **Required capabilities** - Must match original model's capabilities
3. **Provider diversity** - Prefer different providers

14.5 Model Rejection Statistics

Location: Admin Dashboard → Analytics → Model Stats

View per-model: - Total requests and rejections - Rejection rate percentage - Breakdown by rejection type - Fallback success rate

14.6 Database Tables

Table	Purpose
`provider_rejections`	Track all rejections with fallback chain
`rejection_patterns`	Learn patterns for smarter fallback
`user_rejection_notifications`	User-facing notifications
`model_rejection_stats`	Per-model statistics

14.7 Configuration

Setting	Default	Description
`MAX_FALLBACK_ATTEMPTS`	3	Maximum fallback tries per request
`MIN_MODELS_FOR_TASK`	2	Minimum capable models required

See Provider Rejection Handling Documentation for full details.

14.8 Rejection Analytics

Location: Admin Dashboard → Analytics → Rejections

Comprehensive analytics on AI provider rejections to inform ethics policy updates.

Summary Cards

- **Total Rejections (30d)** - All rejections in last 30 days
- **Fallback Success Rate** - Percentage resolved by fallback models
- **Rejected to User** - Requests that couldn't be completed
- **Flagged Keywords** - Keywords marked for policy review

By Provider Tab

Column	Description
Provider	Provider ID (openai, anthropic, google, etc.)
Rejections	Total rejections in period
Models	Number of models affected
Unique Prompts	Distinct prompts rejected
Fallback Rate	Success rate of fallback attempts
Rejected to User	Requests that failed all fallbacks
Types	Rejection types (content_policy, safety_filter, etc.)

Violation Keywords Tab Track which keywords trigger rejections:

Column	Description
Keyword	The triggering keyword
Category	violence, security, controlled, etc.
Occurrences	Times keyword appeared in rejected prompts
Provider Breakdown	Rejections per provider for this keyword
Status	Flagged, Pre-filter added, or Monitoring

Actions: - **Flag for Review** - Mark keyword for policy consideration - **Add Pre-Filter** - Block prompts containing keyword before sending to AI

Flagged Prompts Tab View full content of rejected prompts for investigation:

- Full prompt text (for policy review only)
- Detected violation keywords
- Model and provider that rejected
- Number of times this prompt pattern was rejected
- Suggested actions: Add Pre-Filter, Dismiss

Policy Review Tab Recommendations based on rejection patterns:

- High-frequency rejection patterns
- Suggested pre-filters to add
- Keywords to consider blocking

Database Tables

Table	Purpose
`rejection_analytics`	Daily aggregated stats by model/provider/mode
`rejection_keyword_stats`	Per-keyword occurrence and rejection counts
`rejected_prompt_archive`	Full prompt content for flagged reviews

Using Analytics to Update Ethics Policy

1. **Monitor** → Watch the Keywords tab for high-occurrence terms
2. **Flag** → Mark suspicious keywords for review
3. **Investigate** → View full prompts in Flagged Prompts tab
4. **Decide** → Add pre-filter, add warning, or dismiss
5. **Implement** → Update RADIANT ethics policy to pre-filter prompts

---

15. Security & Compliance

15.1 Security Dashboard

Navigate to **Security** to monitor:

```
Security Dashboard                          Threat Level: Low


Active Threats:     0
Failed Logins:      23 (last 24h)
Suspicious IPs:     2 blocked
MFA Adoption:       94%

Recent Alerts:

   Unusual login location - user@acme.com (2h ago)
   Resolved: Brute force attempt blocked (5h ago)
   Resolved: API key rotated - tenant xyz (1d ago)
```

13.2 Anomaly Detection

Automatic detection of:

- Impossible travel (geographic anomalies)
- Session hijacking attempts
- Brute force attacks
- Unusual API patterns

13.3 Compliance Reports

Navigate to **Compliance** to generate:

Framework	Description
SOC 2	Service organization controls
HIPAA	Healthcare data protection
GDPR	EU data protection
ISO 27001	Information security

13.4 Generating Reports

1. Click **"Generate Report"**
2. Select framework
3. Choose date range
4. Select metrics to include
5. Generate PDF/CSV

---

14. Cost Analytics

14.1 Cost Dashboard

Navigate to **Cost** to view:

```
Cost Analytics                              Period: Last 30 Days


Total Spend:        $12,456.78              (+12% vs last month)
Projected:          $14,200.00              (this month)

By Provider:
   OpenAI:         $6,234.56 (50%)
   Anthropic:      $3,456.78 (28%)
   Self-hosted:    $1,234.56 (10%)
   Other:          $1,530.88 (12%)

AI Recommendations:
  Switch 23% of GPT-4 calls to GPT-4-mini (save $890/mo)
  Enable caching for repeated queries (save $340/mo)
```

14.2 Cost Alerts

Configure alerts:

- Daily budget exceeded
- Weekly spend spike
- Per-tenant limits
- Per-model thresholds

14.3 Cost Optimization

Review AI-powered recommendations:

1. Navigate to **Cost → Insights**
2. Review suggestions
3. Click **"Apply"** to implement (requires approval)
4. Track savings over time

---

15. Revenue Analytics

15.1 Revenue Dashboard

Navigate to **Revenue** to view gross revenue, COGS, and profit:

```
Revenue Analytics                          Period: Last 30 Days


Gross Revenue:      $29,450.00             (+12.5% vs last period)
Total COGS:         $14,150.00
Gross Profit:       $15,300.00             (+15.2% vs last period)
Gross Margin:       51.95%

Revenue Breakdown:
  Subscriptions:        $15,000.00 (50.9%)
  Credit Purchases:     $2,500.00 (8.5%)
  AI Markup (External): $8,750.00 (29.7%)
  AI Markup (Self-Hosted): $3,200.00 (10.9%)

 Note: Marketing, sales, G&A costs not included (COGS only)
```

15.2 Cost Breakdown (COGS)

View infrastructure and provider costs:

Category	Description	Example Services
AWS Compute	Compute infrastructure	EC2, SageMaker, Lambda
AWS Storage	Storage services	S3, EBS
AWS Network	Data transfer	API Gateway, CloudFront
AWS Database	Database services	Aurora, DynamoDB
External AI	Provider costs	OpenAI, Anthropic APIs
Platform Fees	Payment processing	Stripe fees

15.3 Revenue by Model

View per-model profitability:

Model	Provider Cost	Customer Charge	Markup	Requests
gpt-4o	$500.00	$650.00	30%	12,345
claude-3.5-sonnet	$300.00	$390.00	30%	8,901
Self-hosted Llama	$100.00	$175.00	75%	45,678

15.4 Accounting Exports

Export revenue data for accounting software:

1. Click **Export** dropdown
2. Select format:
   - **CSV**: Summary for spreadsheets
   - **JSON**: Full details for integrations
   - **QuickBooks IIF**: Direct import to QuickBooks
   - **Xero CSV**: Import to Xero
   - **Sage CSV**: Import to Sage
3. Configure date range
4. Download file

QuickBooks Integration: - Import via File → Utilities → Import → IIF Files - Creates General Journal entries - Requires matching account names

See Revenue Analytics Documentation for full details.

---

16. SaaS Metrics Dashboard

16.1 Overview

Navigate to **SaaS Metrics** for a comprehensive view of business health:

```
SaaS Metrics Dashboard                    Period: Last 30 Days

MRR: $89,500        ARR: $1,074,000      Gross Margin: 53.1%
+12.5%              +15.2%                        53%

Customers: 342      Churn: 2.3%          LTV:CAC: 6.98x
+5.8%                 Target <2%          Healthy
```

16.2 Key Metrics

Metric	Description	Healthy Range
MRR	Monthly Recurring Revenue	Growing month-over-month
ARR	Annual Recurring Revenue	MRR $\times$ 12
Gross Margin	(Revenue - COGS) / Revenue	$> 50\%$
Churn Rate	Customers lost / Total	$< 3\%$
LTV:CAC	Lifetime Value / Acquisition Cost	$> 3{:}1$

16.3 Dashboard Tabs

1. **Overview**: Revenue trends, top tenants, cost breakdown
2. **Revenue**: MRR movement, revenue by product/tier
3. **Costs**: Cost distribution, COGS breakdown
4. **Customers**: Growth trends, churn analysis
5. **Models**: Per-model profitability analysis

16.4 Exporting Reports

Export data for Excel or accounting:

1. Click **Export** dropdown
2. Select format:
   - **Excel (CSV)**: Full metrics in spreadsheet format
   - **JSON**: Structured data for integrations
3. File downloads with period and date in filename

Export includes: - Revenue summary (MRR, ARR, Gross Profit) - Cost breakdown by category - Customer metrics (total, new, churned) - Unit economics (ARPU, LTV, CAC) - Top tenants with details - Model performance metrics - Daily trend data

See SaaS Metrics Dashboard Documentation for full details.

---

17. A/B Testing & Experiments

16.1 Experiment Dashboard

Navigate to **Experiments** to manage:

Experiment	Status	Variants	Sample Size
Model routing v2	Running	3	45,234
Prompt optimization	Running	2	12,456
Temperature test	Completed	4	89,123

15.2 Creating an Experiment

1. Click **"+ New Experiment"**
2. Configure:
   - **Name**: Descriptive name
   - **Hypothesis**: What you're testing
   - **Variants**: Control + treatments
   - **Traffic Split**: Percentage per variant
   - **Success Metric**: What to measure
3. Set targeting rules
4. Start experiment

15.3 Statistical Analysis

View results with:

- Conversion rates per variant
- Statistical significance (p-value)
- Confidence intervals
- Sample size recommendations

---

16. Audit & Monitoring

16.1 Audit Logs

Navigate to **Audit** to view all actions:

Column	Description
Timestamp	When action occurred
Actor	Who performed action
Action	What was done
Resource	What was affected
IP Address	Source IP
Details	Additional context

Column	Description

16.2 Log Filtering

Filter by:

- Date range
- Actor (user/admin)
- Action type
- Resource type
- Severity level

16.3 Log Export

Export logs for compliance:

1. Set filter criteria
2. Click **"Export"**
3. Choose format (CSV/JSON)
4. Download file

16.4 Real-Time Monitoring

Navigate to **Monitoring** for:

- Live request stream
- Error rate graphs
- Latency percentiles
- Active users count

---

17. Database Migrations

17.1 Migration Workflow

RADIANT uses dual-admin approval for production migrations:

1. **Submit**: Admin submits migration
2. **Review**: Second admin reviews
3. **Approve**: Second admin approves
4. **Execute**: Migration runs
5. **Verify**: Automatic verification

17.2 Pending Migrations

Navigate to **Migrations** to see:

```
Database Migrations


Pending Approval:

  #045 - Add user preferences table
  Submitted by: alice@company.com (2 hours ago)
  [View SQL]  [Approve]  [Reject]


Recent Migrations:
  #044 - Cost tracking tables (applied 2024-12-24)
  #043 - Experiment framework (applied 2024-12-20)
  #042 - Security anomalies (applied 2024-12-15)
```

17.3 Approving Migrations

1. Review the SQL in **"View SQL"**
2. Check for potential issues
3. Click **"Approve"** or **"Reject"**
4. Add comment explaining decision

---

18. API Management

18.1 API Keys

Manage platform API keys:

1. Navigate to **Settings → API Keys**
2. View existing keys
3. Create new keys with scopes
4. Revoke compromised keys

18.2 Rate Limiting

Configure rate limits:

Level	Default	Configurable
Global	10,000/min	Yes
Per-Tenant	1,000/min	Yes
Per-User	100/min	Yes
Per-Key	60/min	Yes

18.3 Webhooks

Configure outgoing webhooks:

1. Navigate to **Settings → Webhooks**
2. Add webhook URL
3. Select events to send
4. Test webhook
5. Enable webhook

---

19. Troubleshooting

19.1 Common Issues

High Error Rate

1. Check **Providers** for unhealthy providers
2. Review **Audit** logs for patterns
3. Check **Monitoring** for load spikes
4. Verify API key validity

Slow Response Times

1. Check provider latency in **Providers**
2. Review model selection in **Orchestration**
3. Check for cold-start issues (self-hosted)
4. Verify database performance

Authentication Failures

1. Check user status in **Users**
2. Verify MFA configuration
3. Review **Audit** logs for login attempts
4. Check for IP blocks in **Security**

19.2 Support Resources

Resource	Description
Documentation	This guide + online docs
Status Page	status.radiant.example.com
Support Email	support@radiant.example.com
Emergency	+1-555-RADIANT

19.3 Log Locations

Service	Log Group
API Gateway	/aws/apigateway/radiant
Lambda	/aws/lambda/radiant-*
Admin Dashboard	/aws/cloudfront/admin
Database	/aws/rds/cluster/radiant

---

20. Delight System Administration

The Delight System provides personality, achievements, and engagement features for Think Tank users.

20.1 Accessing Delight Admin

Navigate to **Think Tank → Delight** in the admin sidebar.

20.2 Dashboard Overview

The Delight dashboard shows:

Metric	Description
Messages Shown	Total delight messages displayed
Achievements Unlocked	Total achievements earned by users
Easter Eggs Found	Hidden features discovered
Active Users	Users with Delight enabled

20.3 Managing Categories

Toggle entire categories on/off:

Category	Purpose
Domain Loading	Messages while loading domain expertise
Domain Transition	Messages when switching topics
Time Awareness	Time-of-day contextual messages
Model Dynamics	Messages about AI consensus/disagreement
Complexity Signals	Feedback on query complexity
Synthesis Quality	Post-response quality indicators
Achievements	Milestone celebrations
Wellbeing	Break/health reminders
Easter Eggs	Hidden features
Sounds	Audio feedback

20.4 Managing Messages

- **Create**: Add new delight messages with targeting options
- **Edit**: Modify text, triggers, and display settings
- **Delete**: Remove messages (soft delete)
- **Toggle**: Enable/disable individual messages

Message Targeting Options

Option	Values
Injection Point	pre_execution, during_execution, post_execution
Trigger Type	domain_loading, time_aware, model_dynamics, etc.
Domain Families	science, humanities, creative, technical, etc.
Time Contexts	morning, afternoon, evening, night, weekend
Display Style	subtle, moderate, expressive

20.5 Statistics Dashboard

Access detailed usage statistics at **Delight → View Statistics**:

- **Weekly Trends**: 12-week activity history
- **Top Messages**: Most-shown messages with engagement data
- **Achievement Stats**: Unlock rates, time-to-unlock averages
- **Easter Egg Stats**: Discovery rates by egg
- **User Engagement**: Leaderboard by achievement points

20.6 Managing Achievements

Configure achievement unlock criteria:

Setting	Description
Threshold	Number required to unlock
Rarity	common, uncommon, rare, epic, legendary
Points	Score value for leaderboards
Hidden	Only visible after unlock

20.7 Managing Easter Eggs

Configure hidden features:

Setting	Description
Trigger Type	key_sequence, text_input, time_based, random
Trigger Value	The activation pattern
Effect Type	mode_change, visual_transform, sound_play

Setting	Description
Duration	How long the effect lasts

20.8 API Endpoints

Endpoint	Method	Description
`/api/admin/delight/dashboard`	GET	Dashboard data
`/api/admin/delight/statistics`	GET	Detailed statistics
`/api/admin/delight/categories/:id`	PATCH	Toggle category
`/api/admin/delight/messages`	POST	Create message
`/api/admin/delight/messages/:id`	PUT/DELETE	Update/delete
`/api/admin/delight/user-engagement`	GET	User leaderboard

Appendix: Quick Reference

Keyboard Shortcuts

Shortcut	Action
`G + D`	Go to Dashboard
`G + T`	Go to Tenants
`G + M`	Go to Models
`G + B`	Go to Billing
`G + A`	Go to Audit
`?`	Show shortcuts

Status Indicators

Icon	Meaning
	Healthy/Success
	Warning
	Error/Failed
	In Progress
	Disabled/Pending

Document Version: 4.18.1 Last Updated: December 2024

Think Tank - Administrator Guide

Configuration and administration of Think Tank AI features

Version: 3.2.0 | Platform: RADIANT 4.18.3 Last Updated: December 2024

Overview

This guide covers administrative features specific to **Think Tank**, the consumer-facing AI assistant platform. For platform-level administration (tenants, billing, infrastructure), see RADIANT-ADMIN-GUIDE.md.

Table of Contents

---

1. Think Tank Admin Features

Location: Admin Dashboard → Think Tank

Think Tank admin features are accessible from the Think Tank section of the Admin Dashboard.

Available Sections

Section	Purpose
My Rules	User memory rules configuration
Delight	Personality and feedback system
Brain Plans	AGI planning visibility
Pre-Prompts	Pre-prompt template management

Section	Purpose
Domains	Domain taxonomy configuration

---

2. User Rules System

Location: Admin Dashboard → Think Tank → My Rules

Users can create personal rules that guide how AI responds to them.

2.1 Rule Types

Type	Description	Example
`instruction`	How to respond	"Always explain in simple terms"
`preference`	User preferences	"I prefer detailed explanations"
`context`	Background info	"I'm a software developer"
`restriction`	Things to avoid	"Never suggest proprietary solutions"

2.2 Memory Categories

Rules are categorized hierarchically:

Category	Subcategories
`instruction`	format, tone, source
`preference`	style, detail
`context`	personal, work, project
`knowledge`	fact, definition, procedure
`constraint`	topic, privacy, safety
`goal`	learning, productivity

2.3 Preset Rules

20+ pre-seeded rule templates across 7 categories that users can add with one click.

2.4 Admin Configuration

Admins can: - View all preset rules - Enable/disable preset categories - Add new preset rules - Set default rules for new users

See User Rules System Documentation for full details.

---

3. Delight System

Location: Admin Dashboard → Think Tank → Delight

The Delight System adds personality, humor, and engaging feedback to AI interactions.

3.1 Features

- **Loading Messages** - Entertaining messages while AI thinks
- **Step Updates** - Progress messages during plan execution
- **Achievements** - Reward milestones (first query, streaks, etc.)
- **Easter Eggs** - Hidden delights for engaged users
- **Wellbeing Nudges** - Gentle reminders for breaks

3.2 Admin Controls

Setting	Description
Enable/Disable	Turn delight system on/off
Message Categories	Enable specific message types
Achievement System	Configure achievement criteria
Easter Eggs	Manage hidden surprises

3.3 Message Types

- Pre-execution messages
- During-execution messages

- Post-execution messages
- Mode-specific messages (coding, creative, research, etc.)

---

4. Brain Plan Viewer

Location: Think Tank → (visible during AI responses)

The Brain Plan Viewer shows users the AGI's plan for solving their prompt.

4.1 What Users See

- **Orchestration Mode** - thinking, coding, creative, research, etc.
- **Domain Detection** - Field, domain, subspecialty, confidence
- **Model Selection** - Which model was chosen and why
- **Step Progress** - Real-time step execution status
- **Timing Estimates** - Expected duration

4.2 Admin Configuration

Setting	Description
Show Plan	Whether to show plan to users
Detail Level	minimal, standard, detailed
Show Costs	Display cost estimates
Show Models	Display model names

---

5. Pre-Prompt Learning

Location: Admin Dashboard → Think Tank → Pre-Prompts

The pre-prompt system selects and learns optimal prompts for different contexts.

5.1 How It Works

1. System selects pre-prompt template based on context
2. User provides feedback on response quality
3. System learns which pre-prompts work best
4. Future selections are optimized

5.2 Admin Features

- View all pre-prompt templates
- See success rates per template
- Adjust learning parameters
- Create new templates

---

6. Domain Taxonomy

Location: Admin Dashboard → Think Tank → Domains

The domain taxonomy helps the AI understand what field/domain a query belongs to.

6.1 Hierarchy

- **Fields** - Top level (e.g., Medicine, Law, Technology)
- **Domains** - Mid level (e.g., Cardiology, Contract Law)
- **Subspecialties** - Specific areas (e.g., Electrophysiology)

6.2 Admin Features

- Add/edit domains
- Configure model proficiencies per domain
- View domain detection accuracy
- Adjust confidence thresholds

---

7. Rejection Notifications

Location: Think Tank → Bell Icon (user view)

When AI providers reject prompts, users are notified with explanations.

7.1 User Experience

- Bell icon shows unread count
- Panel slides out with all notifications
- Each shows: what happened, why, suggested actions
- Resolution status (fallback succeeded, rejected, etc.)

7.2 Suggested Actions

- Rephrase request
- Remove sensitive content
- Try different mode
- Contact administrator

See Provider Rejection Handling Documentation for full details.

---

8. Canvas & Artifacts

Think Tank's canvas feature for interactive content creation.

8.1 Artifact Types

- Code blocks (with execution)
- Documents
- Diagrams
- Data visualizations

8.2 Admin Configuration

- Enable/disable artifact types
- Set size limits
- Configure execution sandboxes

---

9. Collaboration Features

Multi-user collaboration in Think Tank.

9.1 Features

- Shared conversations
- Real-time co-editing
- Team workspaces
- Permission management

9.2 Admin Configuration

- Enable/disable collaboration
- Set sharing defaults
- Configure team limits

Related Documentation

- RADIANT Admin Guide - Platform administration
- Think Tank User Guide - End user guide
- User Rules System - Memory rules details
- Provider Rejection Handling - Rejection system
- AI Ethics Standards - Ethics framework

Deployer Admin Guide

RADIANT Deployer - Administrator Guide

Complete guide for deploying and managing RADIANT infrastructure using the Swift Deployer App

Version: 4.18.1 | Last Updated: December 2024

Table of Contents

---

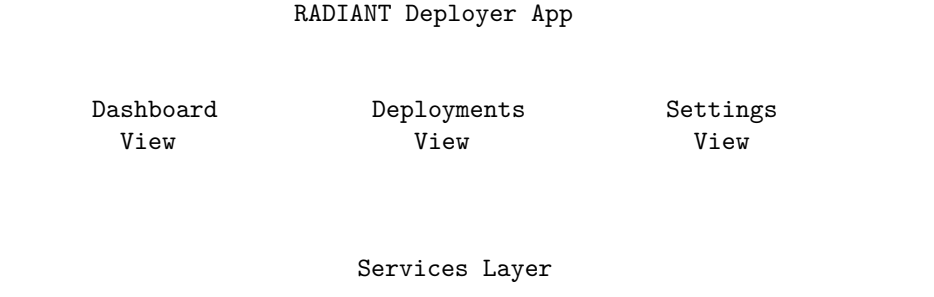
1. Introduction

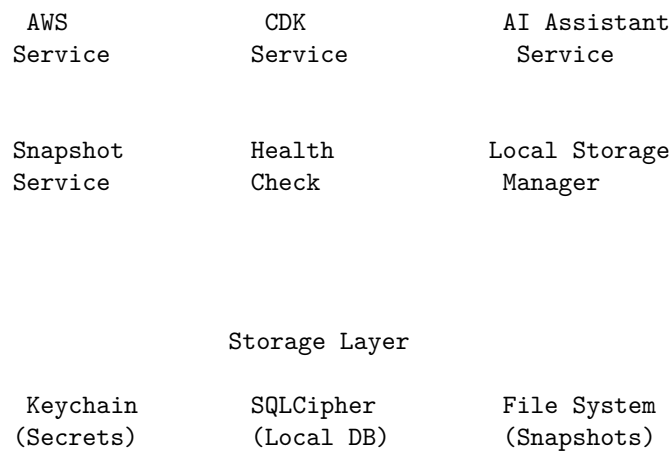
1.1 What is RADIANT Deployer?

RADIANT Deployer is a native macOS application that provides a complete deployment management solution for the RADIANT platform. It offers:

- **One-Click Deployments**: Deploy entire infrastructure stacks with a single click
- **AI-Powered Assistance**: Claude-powered assistant for deployment guidance
- **Snapshot Management**: Create and restore deployment snapshots
- **Multi-Region Support**: Deploy across multiple AWS regions
- **Health Monitoring**: Real-time health checks and status monitoring
- **Secure Credential Storage**: Keychain-integrated credential management

1.2 Architecture





1.3 Key Features

Feature	Description
Deployment Wizard	Step-by-step guided deployment process
Lock-Step Mode	Ensures component version consistency
Automatic Rollback	Reverts failed deployments automatically
Offline Mode	Core functionality works without internet
Audit Logging	Complete deployment history tracking

Deployment Wizard Explained The Deployment Wizard breaks down the complex AWS infrastructure deployment into manageable steps. Instead of manually running CDK commands and configuring services, the wizard:

1. **Validates Prerequisites**: Checks that all required software (Node.js, AWS CLI, CDK) is installed and properly configured before starting
2. **Guides Configuration**: Walks you through selecting environment (dev/staging/prod), tier (1-5), and region settings with explanations for each option
3. **Shows Progress**: Displays real-time progress as each CloudFormation stack deploys, with estimated time remaining
4. **Handles Errors**: If any step fails, provides clear error messages and suggested remediation steps

Lock-Step Mode Explained Lock-Step Mode prevents version mismatches between RADIANT components that could cause compatibility issues:

- **What it does**: Ensures the Admin Dashboard, Lambda functions, and database schema are all on compatible versions
- **Why it matters**: A v4.18 Lambda function might expect database columns that don't exist in a v4.17 schema, causing runtime errors
- **How it works**: Before deployment, the system checks version numbers across all components and blocks deployment if drift exceeds the configured maximum (default: 1 minor version)
- **When to disable**: Only disable during development when testing individual component changes

Automatic Rollback Explained Automatic Rollback protects your production environment by reverting failed deployments:

- **Trigger conditions**: Activates when any deployment phase fails (CDK deploy error, health check failure, migration error)
- **Rollback process**: Restores the pre-deployment snapshot, which includes database state, Lambda code, and configuration
- **Recovery time**: Typically 5-10 minutes depending on the size of changes being reverted
- **Notification**: Sends alerts via configured channels (email, Slack) when rollback occurs

Offline Mode Explained Offline Mode allows essential operations when internet connectivity is unavailable:

- **Available offline**: Viewing deployment history, browsing local snapshots, reviewing configuration, accessing cached documentation
- **Requires internet**: Deploying to AWS, health checks, AI assistant queries, credential validation
- **Data sync**: When connectivity returns, local changes sync automatically with the cloud state

Audit Logging Explained Every action in the Deployer is logged for compliance and troubleshooting:

- **What's logged**: User identity, timestamp, action type, parameters, outcome, duration
- **Storage**: Logs stored locally in SQLCipher database and optionally synced to CloudWatch
- **Retention**: Local logs kept for 90 days by default (configurable)
- **Export**: Logs can be exported to CSV/JSON for compliance audits

---

2. System Requirements

2.1 Hardware Requirements

Component	Minimum	Recommended
macOS Version	13.0 (Ventura)	14.0+ (Sonoma)
Processor	Apple Silicon or Intel	Apple Silicon M1+
Memory	8 GB RAM	16 GB RAM
Storage	2 GB free	10 GB free
Display	1280x800	1440x900+

Why These Requirements? macOS 13.0+: Required for Swift 5.9 runtime and modern SwiftUI features. Older versions lack required system APIs for secure keychain access and modern networking.

Apple Silicon Recommended: While Intel Macs are supported, Apple Silicon provides 2-3x faster CDK synthesis and compilation times. The app is built as a universal binary supporting both architectures.

8 GB RAM Minimum: CDK synthesis loads the entire infrastructure definition into memory. Complex deployments (Tier 3+) with many resources may require more memory. With 8 GB, you may experience slowdowns during synthesis.

16 GB RAM Recommended: Allows comfortable multitasking while deployments run in the background. Essential if you're also running Docker, IDEs, or other development tools.

2 GB Storage Minimum: Covers the application itself (~200 MB), local database (~50 MB), and several snapshots. However, snapshots can grow large over time.

10 GB Storage Recommended: Provides room for multiple deployment snapshots, comprehensive logs, and CDK cache. Each full snapshot can be 100-500 MB depending on your deployment size.

2.2 Software Requirements

Software	Version	Purpose	Installation
Xcode	15.0+	Swift runtime (Command Line Tools sufficient)	`xcode-select --install`
AWS CLI	2.x	AWS operations	`brew install awscli`
Node.js	20.x LTS	CDK operations	`brew install node@20`
AWS CDK	2.120+	Infrastructure deployment	`npm install -g aws-cdk`

Software	Version	Purpose	Installation
pnpm	8.x+	Package management	`npm install -g pnpm`

Software Explained Xcode Command Line Tools: Provides the Swift runtime and compiler. You don't need the full Xcode IDE - just the command line tools (4 GB vs 12 GB download). The Deployer uses Swift for its native performance and seamless macOS integration.

AWS CLI v2: The official AWS command-line interface. Used internally by the Deployer to execute AWS operations, assume IAM roles, and query service status. Version 2 is required for SSO support and improved credential handling.

Node.js 20 LTS: Required to run the AWS CDK, which is written in TypeScript. LTS (Long Term Support) versions receive security updates for 30 months. The Deployer manages Node.js processes internally during CDK operations.

AWS CDK 2.120+: The Cloud Development Kit that defines RADIANT's infrastructure as TypeScript code. Version 2.120+ includes critical bug fixes for Aurora PostgreSQL and Lambda layer handling. The CDK synthesizes your infrastructure into CloudFormation templates.

pnpm 8.x: A fast, disk-efficient package manager used to install CDK dependencies. Chosen over npm for its superior handling of monorepo workspaces and 2-3x faster installation times.

2.3 AWS Requirements

Requirement	Details	How to Verify
AWS Account	Active account with billing enabled	Check AWS Console billing page
IAM User	AdministratorAccess or equivalent	`aws sts get-caller-identity`
Regions	Access to us-east-1 (required) + additional regions	`aws ec2 describe-regions`
Service Quotas	Default quotas sufficient for Tier 1-2	AWS Service Quotas console

AWS Requirements Explained Active AWS Account with Billing: RADIANT deploys real AWS resources that incur costs. Billing must be enabled

and a valid payment method on file. New accounts have a $5 pending charge verification. Free tier covers some resources for 12 months but won't cover all RADIANT components.

IAM User with AdministratorAccess: The Deployer needs broad permissions to create and manage resources across many AWS services. For production, you can use a scoped-down policy (see Appendix A), but AdministratorAccess is recommended for initial setup to avoid permission errors.

Required Permissions Include: - CloudFormation (stack operations) - Lambda (function management) - API Gateway (REST API setup) - RDS/Aurora (database provisioning) - Cognito (user pool management) - S3 (storage buckets) - IAM (role creation) - SSM (parameter storage) - Secrets Manager (credential storage) - CloudWatch (logging and monitoring)

us-east-1 Required: This region is required even if you deploy to other regions because: - ACM certificates for CloudFront must be in us-east-1 - Some global services (IAM, Route 53) operate from us-east-1 - CDK bootstrap resources are region-specific

Service Quotas: Default AWS quotas are sufficient for Tier 1-2 deployments. Tier 3+ may require quota increases for: - Lambda concurrent executions (default: 1,000) - RDS instances (default: 40) - VPC Elastic IPs (default: 5)

To request quota increases: AWS Console > Service Quotas > Select service > Request increase

---

3. Installation

3.1 Download and Install

Option A: Pre-built Application (Recommended)

1. Download the latest release from GitHub Releases
2. Drag `RadiantDeployer.app` to `/Applications`
3. Right-click and select "Open" (first launch only)
4. Grant necessary permissions when prompted

Option B: Build from Source

```
# Clone the repository
git clone https://github.com/your-org/radiant.git
cd radiant/apps/swift-deployer

# Build the application
swift build -c release
```

```
# Run the application
swift run RadiantDeployer
```

3.2 Initial Permissions

The app requires the following permissions:

Permission	Purpose	How to Grant
Keychain Access	Store AWS credentials securely	Approve on first credential save
Network Access	Connect to AWS and AI services	Approve in System Settings
File Access	Save snapshots and logs	Approve when prompted

3.3 Verify Installation

1. Launch RadiantDeployer
2. Navigate to **Settings → About**
3. Verify version shows `4.18.1`
4. Check all services show green status

---

4. First-Time Setup

4.1 Setup Wizard

On first launch, the Setup Wizard guides you through:

```
                    Setup Wizard


Step 1: Welcome                      Complete
Step 2: AWS Credentials              In Progress
Step 3: Environment Configuration    Pending
Step 4: AI Assistant Setup           Pending
Step 5: Verification                 Pending
```

4.2 AWS Credentials Setup

1. Click **"Add AWS Credentials"**
2. Enter your credentials:

- **Name**: Descriptive name (e.g., "Production Account")
- **Access Key ID**: `AKIA...` (20 characters)
- **Secret Access Key**: Your secret key (40 characters)
- **Region**: Primary region (e.g., `us-east-1`)

3. Click **"Validate"** to test connectivity
4. Click **"Save"** to store securely in Keychain

4.3 Environment Configuration

Configure your deployment environment:

Setting	Description	Default
Environment	`dev`, `staging`, or `prod`	`dev`
Tier	Infrastructure tier (1-5)	`1`
Domain	Your domain name	Required for Tier 2+
Stack Prefix	CDK stack name prefix	`radiant`

Environment Types Explained **Development (dev)**: For active development and testing. Features relaxed security settings, smaller instance sizes, and no deletion protection. Data can be reset freely. Cost: ~$50-150/month for Tier 1.

Staging (staging): Pre-production environment that mirrors production configuration. Use this to test deployments before going live. Same security as production but can use smaller instances. Cost: ~60% of production.

Production (prod): Live environment serving real users. Includes deletion protection, multi-AZ deployments, automated backups, and enhanced monitoring. Never deploy untested changes directly to production.

Infrastructure Tiers Explained

Tier	Name	Monthly Cost	Use Case	Resources
1	SEED	$50-150	Development, POC	Single-AZ, t3.small instances, 20GB storage
2	STARTUP	$200-400	Small production	Multi-AZ database, WAF, basic monitoring
3	GROWTH	$1,000-2,500	Medium production	Self-hosted models, HIPAA compliance, enhanced security

Tier	Name	Monthly Cost	Use Case	Resources
4	SCALE	$4,000-8,000	Large production	Multi-region, global database, dedicated instances
5	ENTERPRISE	$35,000-35,000	Enterprise	Custom SLAs, dedicated support, on-premise options

Choosing the Right Tier: Start with Tier 1 for development. Move to Tier 2 when you have paying customers. Tier 3+ is for organizations with compliance requirements or high traffic.

Domain Configuration For Tier 2+, you need a custom domain:

1. **Register a domain** in Route 53 or transfer an existing domain
2. **Create a hosted zone** in Route 53 for your domain
3. **Request an ACM certificate** in us-east-1 for `*.yourdomain.com`
4. **Validate the certificate** via DNS (automatic if using Route 53)

The Deployer will configure: - `api.yourdomain.com` - API Gateway endpoint - `admin.yourdomain.com` - Admin Dashboard - `app.yourdomain.com` - Think Tank application

4.4 AI Assistant Setup (Optional)

Enable the Claude-powered AI assistant:

1. Navigate to **Settings → AI Assistant**
2. Enter your Anthropic API key
3. Select response style:
   - **Concise**: Brief, action-focused responses
   - **Detailed**: In-depth explanations
   - **Tutorial**: Step-by-step guidance
4. Test the connection with a sample query

---

5. AWS Credentials Management

5.1 Credential Sets

Manage multiple AWS accounts:

Field	Description	Example
Name	Friendly identifier	"Production"
Access Key ID	AWS access key	`AKIAIOSFODNN7EXAMPLE`
Secret Access Key	AWS secret	(stored encrypted)
Region	Default region	`us-east-1`
Role ARN	Optional assume role	`arn:aws:iam::123:role/deploy`

5.2 Adding Credentials

1. Navigate to **Credentials** tab
2. Click **"+ Add Credential Set"**
3. Fill in the required fields
4. Click **"Validate"** to test
5. Click **"Save"**

5.3 Credential Validation

The app validates:

- Access key format (AKIA prefix, 20 chars)
- Secret key length (40+ chars)
- Region validity
- AWS connectivity (STS GetCallerIdentity)
- Required permissions

5.4 Security Best Practices

Practice	Recommendation
Rotate Keys	Every 90 days
Least Privilege	Use scoped IAM policies
MFA	Enable on AWS account
Audit	Review access logs regularly
Backup	Export credentials securely

5.5 Importing from AWS CLI

```
# The app can import from ~/.aws/credentials
# Navigate to Credentials → Import from AWS CLI
```

---

6. Deployment Operations

6.1 Deployment Dashboard

The main dashboard shows:

```
Environment: dev          Status:   Healthy
Version: 4.18.1           Last Deploy: 2024-12-25 10:30:00


   Deploy           Rollback         Settings
   [Button]         [Button]         [Button]


Recent Deployments:

  2024-12-25 10:30   v4.18.1   prod    Success   4m 32s
  2024-12-24 15:45   v4.18.0   prod    Success   5m 12s
  2024-12-24 09:00   v4.17.0   dev     Success   3m 45s
```

6.2 Starting a Deployment

1. Select target **Environment** (dev/staging/prod)
2. Select **Tier** (1-5)
3. Review deployment plan
4. Click **"Start Deployment"**
5. Monitor progress in real-time

6.3 Deployment Phases

Phase	Duration	Description
1. Validation	~30s	Credential and configuration check
2. Snapshot	~1m	Create pre-deployment backup
3. CDK Synth	~1m	Generate CloudFormation templates
4. CDK Deploy	~10-20m	Deploy infrastructure
5. Migration	~2m	Run database migrations
6. Health Check	~1m	Verify all services
7. Cleanup	~30s	Remove temporary resources

Phase Details Phase 1 - Validation (30 seconds)

Before any changes are made, the system validates: - AWS credentials are valid and not expired - IAM permissions are sufficient for all required operations - Target environment exists or can be created - No conflicting deployments are in progress (deployment lock check) - Required software versions are installed (Node.js, CDK, etc.) - Network connectivity to AWS services

If validation fails, you'll see specific error messages explaining what needs to be fixed.

Phase 2 - Snapshot (1 minute)

A pre-deployment snapshot captures the current state so you can rollback if needed: - Database schema and critical data (not full data backup) - Current Lambda function code versions - SSM Parameter Store values - Current CloudFormation stack states - Configuration files

Snapshots are stored locally and can be managed in the Snapshots tab.

Phase 3 - CDK Synth (1 minute)

The CDK synthesizes TypeScript infrastructure code into CloudFormation templates: - Reads infrastructure definitions from `packages/infrastructure/` - Resolves all construct dependencies - Generates CloudFormation JSON/YAML templates - Calculates resource changes (what will be created/updated/deleted) - Validates templates against AWS CloudFormation rules

You can review the generated templates before proceeding.

Phase 4 - CDK Deploy (10-20 minutes)

The actual AWS resource deployment happens in this phase: - CloudFormation stacks are created or updated in dependency order - Resources are provisioned (databases, Lambda functions, API Gateway, etc.) - IAM roles and policies are configured - Networking (VPC, subnets, security groups) is set up - DNS records are created/updated

This is the longest phase. Progress shows which stack is currently deploying.

Phase 5 - Migration (2 minutes)

Database migrations ensure your schema matches the deployed code: - Connects to Aurora PostgreSQL using Data API - Runs pending migration files in sequence - Creates new tables, columns, indexes as needed - Updates RLS (Row-Level Security) policies - Verifies migration success with integrity checks

Migrations are idempotent - running them twice won't cause issues.

Phase 6 - Health Check (1 minute)

Verifies all deployed services are functioning: - API Gateway responds to health endpoint - Lambda functions can be invoked - Database connections succeed - Cognito user pools are accessible - S3 buckets are reachable - CloudFront distributions are deployed

If health checks fail, automatic rollback is triggered (if enabled).

Phase 7 - Cleanup (30 seconds)

Final cleanup tasks: - Removes temporary files created during deployment - Clears CDK staging buckets of old assets - Updates deployment history in local database - Releases deployment lock - Sends completion notification

6.4 Deployment Progress

```
Deploying to Production


[                ] 65%

Current Phase: CDK Deploy
Stack: Radiant-prod-API (5 of 9)
Elapsed: 8m 23s | Estimated: 4m remaining

  Validation complete
  Snapshot created: snap-20241225-103000
  CDK synthesis complete
  Deploying stacks...
    Radiant-prod-Foundation
    Radiant-prod-Networking
    Radiant-prod-Security
    Radiant-prod-Data
    Radiant-prod-API
    Radiant-prod-Auth
    Radiant-prod-AI
    Radiant-prod-Admin
    Radiant-prod-Monitoring

[Cancel Deployment]
```

6.5 Deployment Settings

Configure deployment behavior:

Setting	Description	Default
Auto-Rollback	Rollback on failure	Enabled
Lock-Step Mode	Require version consistency	Enabled
Max Version Drift	Maximum version difference	1
Approval Required	Require confirmation for prod	Enabled
Notification	Send completion notifications	Enabled

6.6 Operation Timeouts

Operation	Default Timeout	Configurable
CDK Deploy	30 minutes	Yes
Health Check	5 minutes	Yes
Migration	10 minutes	Yes
Snapshot	5 minutes	Yes
Rollback	15 minutes	Yes

---

7. Multi-Region Deployments

7.1 Overview

Deploy RADIANT across multiple AWS regions for:

- **High Availability**: Survive regional outages
- **Low Latency**: Serve users from nearest region
- **Compliance**: Data residency requirements

7.2 Supported Regions

Region	Code	Primary Use
US East (N. Virginia)	`us-east-1`	Primary (required)
US West (Oregon)	`us-west-2`	West coast users
EU (Ireland)	`eu-west-1`	European users
EU (Frankfurt)	`eu-central-1`	GDPR compliance
Asia Pacific (Singapore)	`ap-southeast-1`	APAC users
Asia Pacific (Tokyo)	`ap-northeast-1`	Japanese users

7.3 Adding a Region

1. Navigate to **Multi-Region** tab
2. Click **"Add Region"**
3. Configure:
   - **Region**: Select from available regions
   - **Is Primary**: Set primary region flag
   - **Stack Prefix**: Region-specific prefix
   - **Endpoint**: Custom domain for region
4. Click **"Deploy to Region"**

7.4 Region Consistency

Monitor version consistency across regions:

```
Multi-Region Status                        Consistency:   100%

Region            Version    Status    Last Deploy

us-east-1 (P)     4.18.1       Healthy  2024-12-25 10:30
eu-west-1         4.18.1       Healthy  2024-12-25 10:35
ap-southeast-1    4.18.1       Healthy  2024-12-25 10:40

[Deploy All]  [Check Consistency]  [Sync Versions]
```

---

8. Snapshots & Rollbacks

8.1 Snapshot Types

Type	Description	Retention
Pre-Deploy	Automatic before each deployment	30 days
Manual	User-initiated backup	Until deleted
Scheduled	Periodic backups	Configurable

8.2 Creating a Snapshot

1. Navigate to **Snapshots** tab
2. Click **"Create Snapshot"**
3. Enter description (optional)
4. Select components to include:
   - Database state
   - Configuration
   - Lambda code
   - Infrastructure state
5. Click **"Create"**

8.3 Snapshot Contents

```
Snapshot: snap-20241225-103000
  metadata.json           # Snapshot info
  database/
     schema.sql           # Database schema
     data.sql             # Critical data
  config/
     ssm-parameters.json # SSM parameters
```

```
    secrets.json        # Secret references
lambda/
    functions.zip       # Lambda code packages
infrastructure/
    state.json          # CDK state
```

8.4 Restoring from Snapshot

1. Navigate to **Snapshots** tab
2. Select snapshot to restore
3. Click **"Restore"**
4. Confirm restoration scope:
   - Full restoration
   - Database only
   - Configuration only
5. Click **"Confirm Restore"**

8.5 Automatic Rollback

When enabled, failed deployments automatically:

1. Stop deployment process
2. Identify last known good state
3. Restore from pre-deploy snapshot
4. Verify system health
5. Send notification

---

9. AI Assistant

9.1 Overview

The Claude-powered AI Assistant provides:

- **Deployment Guidance**: Step-by-step help
- **Error Translation**: User-friendly error explanations
- **Troubleshooting**: Problem diagnosis
- **Best Practices**: Recommendations

9.2 Enabling AI Assistant

1. Navigate to **Settings → AI Assistant**
2. Enter Anthropic API key
3. Toggle **"Enable AI Assistant"**
4. Select response style

9.3 Using the Assistant

Access the assistant via:

- **Chat Panel**: Click AI icon in toolbar
- **Contextual Help**: Click "?" on any screen
- **Error Dialogs**: Click "Explain" on errors

9.4 Sample Interactions

```
AI Assistant


You: What does error "AccessDenied: User is not authorized
     to perform dynamodb:CreateTable" mean?

Assistant: This error indicates your AWS credentials don't
have permission to create DynamoDB tables. To fix this:

1. Check your IAM user/role has the required permissions
2. Add the "AmazonDynamoDBFullAccess" managed policy, or
3. Add specific permissions:
   - dynamodb:CreateTable
   - dynamodb:DescribeTable
   - dynamodb:UpdateTable

Would you like me to show the exact IAM policy needed?

[Yes, show policy]  [No, I'll figure it out]
```

9.5 Offline Mode

When offline, the assistant provides:

- Pre-cached common error explanations
- Local troubleshooting guides
- Fallback recommendations

---

10. Package Management

10.1 Package System

RADIANT uses atomic packages for deployment:

```
radiant-4.18.1.pkg
  manifest.json           # Package manifest
  checksums.sha256        # Component checksums
  radiant/                # Radiant components
      infrastructure/
      lambda/
      dashboard/
  thinktank/              # Think Tank components
       api/
       frontend/
```

10.2 Viewing Packages

Navigate to **Packages** tab to see:

- Installed packages
- Available updates
- Package history
- Component versions

10.3 Version Management

Version Type	Format	Example
Radiant	Major.Minor.Patch	4.18.1
Think Tank	Major.Minor.Patch	3.2.0
Package	Combined	4.18.1+3.2.0

10.4 Lock-Step Mode

When enabled:

- All components must have same minor version
- Maximum version drift configurable
- Automatic sync available

---

11. Monitoring & Health Checks

11.1 Health Dashboard

```
  System Health                          Overall:  Healthy

  Service              Status    Latency    Last Check
```

```
API Gateway            Up      45ms       10s ago
Lambda (Router)        Up      120ms      10s ago
Aurora PostgreSQL      Up      12ms       10s ago
DynamoDB               Up      8ms        10s ago
Cognito                Up      85ms       10s ago
S3 Storage             Up      35ms       10s ago
CloudFront             Up      22ms       10s ago
SageMaker (if T3+)     Up      250ms      10s ago

[Refresh]  [Run Full Check]  [View History]
```

11.2 Health Check Types

Check	Frequency	Timeout
Quick	Every 60s	5s
Standard	Every 5m	30s
Deep	Manual/Deploy	2m

11.3 Alerts

Configure alerts for:

- Service degradation
- High latency
- Error rate spikes
- Failed deployments

---

12. Security Features

12.1 Credential Security

Feature	Implementation
Storage	macOS Keychain (encrypted)
Memory	Cleared after use
Transport	TLS 1.3 only
Validation	Format + connectivity check

12.2 Deployment Locks

Prevent concurrent deployments:

```
Deployment Lock: Active
  Acquired: 2024-12-25 10:30:00
  Owner: deployer@example.com
  Environment: production
  Expires: 2024-12-25 11:30:00
```

12.3 Audit Logging

All operations are logged:

```
{
  "timestamp": "2024-12-25T10:30:00Z",
  "operation": "deployment.start",
  "user": "admin@example.com",
  "environment": "production",
  "version": "4.18.1",
  "status": "success",
  "duration_ms": 272000
}
```

12.4 Secret Detection

Pre-commit checks scan for:

- AWS access keys
- API keys
- Passwords
- Private keys

---

13. Troubleshooting

13.1 Common Issues

Deployment Fails at CDK Synth **Symptoms**: Deployment stops at synthesis phase

Solutions: 1. Check Node.js version: `node --version` (need 20.x) 2. Clear CDK cache: `rm -rf cdk.out` 3. Update CDK: `npm update -g aws-cdk` 4. Check TypeScript errors in `packages/infrastructure`

AWS Credentials Invalid **Symptoms**: "Invalid credentials" error

Solutions: 1. Verify access key format (starts with AKIA) 2. Check secret key hasn't expired 3. Verify IAM user is active 4. Test with AWS CLI: `aws sts get-caller-identity`

Health Check Timeout **Symptoms**: Services show unhealthy after deployment

Solutions: 1. Wait 2-3 minutes for cold start 2. Check CloudWatch logs for errors 3. Verify security group rules 4. Check VPC endpoint configuration

13.2 Log Locations

Log Type	Location
App Logs	`~/Library/Logs/RadiantDeployer/`
Deployment Logs	`~/Library/Application Support/RadiantDeployer/deployments/`
AWS Logs	CloudWatch Log Groups

13.3 Getting Help

1. **AI Assistant**: Built-in help
2. **Documentation**: This guide + online docs
3. **Support**: support@radiant.example.com

---

14. Reference

14.1 Keyboard Shortcuts

Shortcut	Action
+ D	Start deployment
+ R	Refresh status
+ S	Create snapshot
+ ,	Open settings
+ ?	Open AI assistant
+ L	View logs

14.2 CLI Commands

```
# Build and run from source
cd apps/swift-deployer
swift build -c release
swift run RadiantDeployer

# Run with specific config
swift run RadiantDeployer --environment prod --tier 3
```

```
# Headless deployment
swift run RadiantDeployer deploy --non-interactive
```

14.3 Environment Variables

Variable	Description
`RADIANT_ENV`	Override environment
`RADIANT_TIER`	Override tier
`RADIANT_DEBUG`	Enable debug logging
`RADIANT_AI_KEY`	Anthropic API key

14.4 File Locations

File	Location
Configuration	`~/Library/Application Support/RadiantDeployer/config.json`
Snapshots	`~/Library/Application Support/RadiantDeployer/snapshots/`
Logs	`~/Library/Logs/RadiantDeployer/`
Database	`~/Library/Application Support/RadiantDeployer/local.db`

Appendix A: IAM Policy Requirements

Minimum IAM permissions for deployment:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "s3:*",
        "lambda:*",
        "apigateway:*",
        "cognito-idp:*",
        "rds:*",
        "dynamodb:*",
        "sqs:*",
        "sns:*",
        "events:*",
```

```
        "logs:*",
        "iam:PassRole",
        "iam:CreateRole",
        "iam:AttachRolePolicy",
        "ssm:*",
        "secretsmanager:*",
        "ecr:*",
        "ecs:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Appendix B: Glossary

Term	Definition
CDK	AWS Cloud Development Kit
Stack	CloudFormation stack deployed by CDK
Snapshot	Point-in-time backup of deployment
Lock-Step	Version consistency enforcement
Tier	Infrastructure sizing level (1-5)

Document Version: 4.18.1 Last Updated: December 2024

PART 3: USER GUIDES

Think Tank User Guide

Think Tank AI - User Guide

Your gateway to 100+ AI models in one place

Version: 3.2.0 (Platform: RADIANT 4.18.1) Last Updated: December 2024

Welcome to Think Tank

Think Tank is your all-in-one AI assistant platform. Access the world's best AI models—GPT-4, Claude, Gemini, and 100+ more—from a single, beautiful interface.

Table of Contents

---

1. Getting Started

1.1 Creating Your Account

1. Visit **thinktank.ai**
2. Click **Get Started Free**
3. Sign up with:
   - Email and password
   - Google account
   - Microsoft account
   - Apple ID
4. Verify your email
5. Complete your profile

1.2 Choosing a Plan

Plan	Price	Best For
Free	$0/month	Trying out Think Tank
Starter	$29/month	Individual creators
Pro	$99/month	Power users
Team	$49/user/month	Small teams

Plan	Price	Best For
Business	$199/user/month	Organizations

1.3 Your First Chat

1. Click **New Chat** or press `Ctrl+N`
2. Type your question or request
3. Press `Enter` or click **Send**
4. Watch as AI responds in real-time

Try these starter prompts: - "Explain quantum computing like I'm 10 years old" - "Write a professional email declining a meeting" - "Help me debug this Python code: [paste code]" - "Create a meal plan for the week"

---

2. Your Dashboard

2.1 Interface Overview

```
Think Tank                      [Search]  [Credits: 450]


Chats                          Chat Area

Starred      Welcome! How can I help you today?
Today
 Chat 1
 Chat 2
Yesterday
 Chat 3


Personas     Type your message...                [Send]
Canvas
Settings
             Model: GPT-4 Turbo  |  Focus: General  |
```

2.2 Sidebar Navigation

Icon	Section	Description
	Chats	All your conversations
	Starred	Important chats

Icon	Section	Description
	Personas	Custom AI personalities
	Canvas	Visual workspace
	Usage	Credit usage stats
	Settings	Account settings

2.3 Quick Actions

Action	Shortcut
New Chat	`Ctrl+N`
Search	`Ctrl+K`
Toggle Sidebar	`Ctrl+B`
Settings	`Ctrl+,`

---

3. Chatting with AI

3.1 Sending Messages

Text Messages: - Type in the input box - Press `Enter` to send - Use `Shift+Enter` for new lines

Attachments: - Click to attach files - Drag & drop images, PDFs, code files - Paste images directly (`Ctrl+V`)

Voice Input: - Click microphone icon - Speak your message - Click again to stop

3.2 Message Actions

Hover over any message to see actions:

Icon	Action	Description
	Copy	Copy message text
	Regenerate	Get a new response
	Edit	Modify your message
/	Rate	Help improve AI
	Pin	Keep message visible
	Delete	Remove message

3.3 Streaming Responses

AI responses stream in real-time. You can: - **Stop**: Click to stop generation - **Continue**: Ask "continue" if response was cut off - **Regenerate**: Get a different response

3.4 Multi-Turn Conversations

Think Tank remembers your conversation context:

```
You: I'm planning a trip to Japan
AI:  That's exciting! When are you planning to visit...

You: What about the food?
AI:  Japanese cuisine is incredible! Based on your trip...
     [AI remembers you're going to Japan]
```

3.5 Code in Chats

Code is automatically syntax-highlighted:

```python
def hello_world():
    print("Hello, Think Tank!")
```

Click **Copy** to copy code blocks, or **Run** for supported languages.

---

4. Choosing Models

4.1 Model Selection

Click the model selector at the bottom of the chat:

```
Select Model

  Favorites
   GPT-4 Turbo          $0.02/msg
   Claude 3 Opus        $0.03/msg

  Recommended
   GPT-4o               $0.01/msg
   Claude 3.5 Sonnet    $0.01/msg
   Gemini 1.5 Pro       $0.01/msg

  Writing
  Coding
  Analysis
  Creative
```

`[View All 100+ Models]`

4.2 Model Categories

Category	Best For	Top Models
General	Everyday tasks	GPT-4o, Claude 3.5
Writing	Content creation	Claude 3 Opus, GPT-4
Coding	Programming help	GPT-4 Turbo, CodeLlama
Analysis	Data & research	Gemini 1.5, Claude 3
Creative	Art & ideas	GPT-4, Mistral Large
Vision	Image understanding	GPT-4V, LLaVA
Fast	Quick responses	GPT-3.5, Claude Instant

Choosing the Right Model **For everyday questions and tasks**: Start with GPT-4o or Claude 3.5 Sonnet. These models offer the best balance of quality, speed, and cost. They handle most tasks excellently including writing, answering questions, brainstorming, and light coding.

For professional writing: Claude 3 Opus excels at long-form content, maintaining consistent tone, and nuanced writing. GPT-4 is also excellent for business documents and creative writing.

For coding and technical work: GPT-4 Turbo has strong coding abilities across many languages. For specialized tasks, consider CodeLlama (open source, good for common languages) or specialized models like DeepSeek Coder.

For data analysis: Gemini 1.5 Pro handles very long documents (up to 1 million tokens) making it ideal for analyzing large datasets or documents. Claude 3 is excellent for nuanced analytical reasoning.

For creative projects: GPT-4 and Mistral Large are both creative and can help with brainstorming, storytelling, and idea generation. They're less constrained in creative contexts.

For image understanding: GPT-4V (Vision) and Claude 3 Vision can analyze images, read text from photos, describe scenes, and answer questions about visual content.

For quick, simple tasks: GPT-3.5 Turbo and Claude Instant are much faster and cheaper. Use them for simple questions, formatting, or when you need instant responses.

4.3 Auto Mode

Let Think Tank choose the best model:

1. Enable **Auto Mode** in settings

2. Our Brain Router analyzes your request
3. Automatically selects optimal model
4. Balances quality, speed, and cost

How Auto Mode Works When you enable Auto Mode, Think Tank's Brain Router analyzes each message you send and selects the best model based on:

- **Task complexity**: Simple questions go to fast models; complex tasks go to powerful models
- **Content type**: Coding questions route to code-specialized models; creative requests to creative models
- **Your history**: Learns your preferences over time and adjusts recommendations
- **Cost efficiency**: Avoids using expensive models when cheaper ones would work equally well
- **Current availability**: Routes around any models experiencing slowdowns

When to use Auto Mode: - You're not sure which model to use - You want to optimize cost without sacrificing quality - You have varied tasks throughout the day - You're new to Think Tank

When to choose manually: - You need a specific model's unique capabilities - You're doing specialized work (e.g., always want Claude for writing) - You're comparing models intentionally - You have strong preferences for certain models

4.4 Model Comparison

Split-screen to compare models:

1. Click **Compare** button
2. Select 2-4 models
3. Send message to all simultaneously
4. See responses side-by-side

---

5. Focus Modes & Personas

5.1 Focus Modes

Pre-configured modes for specific tasks:

Mode	Optimized For
Professional	Business writing, emails
Developer	Code, debugging, architecture
Research	Analysis, citations, accuracy
Creative	Stories, brainstorming

Mode	Optimized For
Learning	Explanations, tutoring
Concise	Brief, direct answers

To switch modes: 1. Click the Focus selector 2. Choose your mode 3. AI adapts its style

Focus Mode Details Professional Mode: The AI adopts a business-appropriate tone. Responses are polished, formal, and suitable for workplace communication. Great for drafting emails, reports, presentations, and client communications. Avoids casual language and ensures professional formatting.

Developer Mode: Optimized for technical work. The AI provides code with proper syntax highlighting, explains technical concepts clearly, suggests best practices, and can help debug issues. Responses include code comments and consider edge cases.

Research Mode: Emphasizes accuracy and thoroughness. The AI cites sources when possible, acknowledges uncertainty, presents multiple perspectives, and structures information logically. Ideal for academic work, fact-checking, and deep analysis.

Creative Mode: Removes constraints on creativity. The AI is more willing to explore unusual ideas, use vivid language, and think outside the box. Perfect for brainstorming, creative writing, storytelling, and generating innovative solutions.

Learning Mode: The AI becomes a patient tutor. Explanations start from basics and build up, concepts are broken into digestible pieces, and the AI checks understanding before moving on. Great for studying new topics.

Concise Mode: Responses are brief and to the point. The AI avoids lengthy explanations and gets straight to the answer. Useful when you need quick facts or are in a hurry.

5.2 Custom Personas

Create your own AI personalities:

1. Go to **Personas** → **Create New**
2. Configure:
   - **Name**: "Marketing Expert"
   - **Personality**: Professional, enthusiastic
   - **Expertise**: Digital marketing, SEO
   - **Style**: Uses bullet points, data-driven
3. Click **Save**

Example Persona:

```
Name: Code Reviewer
Personality: Thorough, constructive
Instructions: Review code for bugs, security issues,
              and best practices. Always suggest
              improvements with examples.
```

5.3 Sharing Personas

- **Public**: Share with all Think Tank users
- **Team**: Share within your organization
- **Private**: Only you can use

---

6. Canvas & Artifacts

6.1 What is Canvas?

Canvas is your visual workspace for complex outputs: - Code files with syntax highlighting - Diagrams and flowcharts - Documents and reports - Data tables - Mind maps

6.2 Creating Artifacts

When AI generates complex content, it appears as an artifact:

```
  business_plan.md

# Business Plan

## Executive Summary
...


[Copy] [Download] [Edit] [Version History]
```

6.3 Artifact Actions

Action	Description
Copy	Copy content to clipboard
Download	Save as file
Edit	Modify directly
Versions	View previous versions
Share	Generate share link
To Canvas	Open in full Canvas view

6.4 Full Canvas Mode

For larger projects:

1. Click **Canvas** in sidebar
2. Create new canvas or open existing
3. Add multiple artifacts
4. Arrange spatially
5. Connect related items

---

7. Collaboration Features

7.1 Sharing Chats

Share any conversation:

1. Click **Share** ( ) on a chat
2. Choose visibility:
   - **Link**: Anyone with link
   - **Team**: Your organization
   - **Private**: Specific people
3. Copy and share the link

7.2 Real-Time Collaboration

Work together on the same chat:

1. Share chat with **Edit** access
2. Multiple users can:
   - Send messages simultaneously
   - See each other's cursors
   - React to messages
3. Changes sync in real-time

7.3 Team Workspaces

For Team and Business plans:

- **Shared Chats**: Team-visible conversations
- **Shared Personas**: Team AI configurations
- **Shared Canvas**: Collaborative workspaces
- **Usage Dashboard**: Team analytics

7.4 Comments & Annotations

Add notes to any message:

1. Hover over message
2. Click **Comment** ( )

3. Add your note
4. Tag teammates with @mention

---

8. Managing Your Account

8.1 Profile Settings

Access via **Settings** → **Profile**:

Setting	Description
Display Name	Your visible name
Email	Login email
Avatar	Profile picture
Language	Interface language
Timezone	For scheduled features

8.2 Preferences

Customize your experience:

Preference	Options
Theme	Light, Dark, System
Font Size	Small, Medium, Large
Default Model	Your preferred model
Auto Mode	Enable/disable
Sound Effects	On/Off
Notifications	Email, Push, None

8.3 Data & Privacy

Control your data:

- **Export Data**: Download all your chats
- **Delete History**: Remove chat history
- **Training Opt-Out**: Exclude from AI training
- **Data Retention**: Set auto-delete period

8.4 Connected Apps

Manage integrations:

- Google Drive
- Dropbox
- Notion

- Slack
- GitHub

---

9. Credits & Billing

9.1 Understanding Credits

Credits are Think Tank's universal currency:

Credit Value	Equivalent
1 credit	$0.01
100 credits	$1.00

9.2 Credit Usage

Different models cost different amounts:

Model	~Cost per Message
GPT-3.5 Turbo	0.5 credits
GPT-4o	1-2 credits
GPT-4 Turbo	2-3 credits
Claude 3.5 Sonnet	1-2 credits
Claude 3 Opus	3-5 credits

Actual cost depends on message length

9.3 Viewing Usage

Check your usage in **Settings** → **Usage**:

```
Credit Usage - December 2024

Balance:        450 credits
Used this month: 1,550 credits
Included:       2,000 credits

[          ] 77% used

By Model:
  GPT-4 Turbo     800 credits (52%)
  Claude 3        500 credits (32%)
  Other           250 credits (16%)
```

9.4 Purchasing Credits

Need more credits?

1. Go to **Settings** → **Billing**
2. Click **Buy Credits**
3. Select amount:
   - 500 credits - $5
   - 1,000 credits - $9 (10% bonus)
   - 5,000 credits - $40 (20% bonus)
4. Complete payment

9.5 Subscription Management

Manage your plan:

- **Upgrade**: Get more features and credits
- **Downgrade**: Switch to lower tier (end of period)
- **Cancel**: Cancel subscription (keep access until end)
- **Invoices**: Download billing history

---

10. Tips & Best Practices

10.1 Writing Better Prompts

Be Specific:

```
"Write about dogs"
"Write a 200-word blog post about training golden
  retriever puppies, focusing on positive reinforcement"
```

Provide Context:

```
"Fix this code"
"Fix this Python code that should sort a list but
  throws an IndexError: [paste code]"
```

Set the Format:

```
"Give me ideas"
"Give me 5 blog post ideas about sustainable living,
  formatted as bullet points with a brief description"
```

10.2 Getting Better Results

Technique	Example
Chain of thought	"Think step by step…"
Role assignment	"Act as a senior developer…"

Technique	Example
Examples	"Here's an example of what I want…"
Constraints	"In 100 words or less…"
Iteration	"Good, but make it more formal"

10.3 Saving Credits

- Use **Auto Mode** for optimal model selection
- Use **GPT-3.5** for simple tasks
- Be concise in your prompts
- Avoid regenerating unnecessarily
- Use **Focus Modes** for specialized tasks

10.4 Organizing Chats

- **Star** important conversations
- **Folders**: Group related chats
- **Tags**: Add searchable labels
- **Search**: Find any past conversation

---

11. Keyboard Shortcuts

11.1 General

Shortcut	Action
`Ctrl+N`	New chat
`Ctrl+K`	Search
`Ctrl+B`	Toggle sidebar
`Ctrl+,`	Settings
`Ctrl+/`	Show shortcuts
`Escape`	Close modal

11.2 Chat

Shortcut	Action
`Enter`	Send message
`Shift+Enter`	New line
`Ctrl+↑`	Edit last message
`Ctrl+Shift+C`	Copy last response
`Ctrl+Shift+R`	Regenerate
`Ctrl+.`	Stop generation

11.3 Navigation

Shortcut	Action
`Alt+↑/↓`	Previous/next chat
`Ctrl+1-9`	Switch to chat 1-9
`Ctrl+Tab`	Cycle tabs

---

12. FAQ

Getting Started

Q: Is Think Tank free? A: Yes! The Free plan includes 50 credits/month. Upgrade for more credits and features.

Q: Which AI model should I use? A: Enable Auto Mode and let us choose, or: - General tasks → GPT-4o or Claude 3.5 Sonnet - Complex analysis → GPT-4 Turbo or Claude 3 Opus - Quick answers → GPT-3.5 Turbo

Q: Can I use Think Tank on mobile? A: Yes! Visit thinktank.ai on any mobile browser, or download our iOS/Android apps.

Credits & Billing

Q: What happens when I run out of credits? A: You can purchase more credits or wait for your monthly refresh (paid plans).

Q: Do unused credits roll over? A: Monthly included credits expire. Purchased credits never expire.

Q: Can I get a refund? A: Contact support within 14 days for subscription refunds.

Privacy & Security

Q: Is my data used to train AI? A: By default, no. You can verify in Settings → Privacy.

Q: Who can see my chats? A: Only you, unless you explicitly share them.

Q: Is my data encrypted? A: Yes, with AES-256 encryption at rest and TLS 1.3 in transit.

Troubleshooting

Q: Why is the AI response slow? A: Complex queries or busy times may cause delays. Try a faster model.

Q: Why did my response get cut off? A: Models have output limits. Type "continue" to get the rest.

Q: I found a bug. How do I report it? A: Click **Help** → **Report Issue** or email support@thinktank.ai.

Need Help?

- **Help Center**: help.thinktank.ai
- **Live Chat**: Click the chat bubble
- **Email**: support@thinktank.ai
- **Twitter**: @ThinkTankAI
- **Discord**: discord.gg/thinktank

Version History

Version	Date	Changes
3.2.0	December 2024	Time Machine, enhanced collaboration, A/B experiments, **Delight System**
3.1.0	November 2024	Canvas improvements, new models
3.0.0	October 2024	Initial release

---

13. Delight System

Think Tank includes a personality system called "Delight" that makes your AI experience more engaging.

13.1 What is Delight?

Delight adds contextual, friendly messages during your conversations:

- **Domain Loading**: "Consulting the fundamental forces…" when you ask physics questions
- **Time Awareness**: "Burning the midnight tokens" during late-night sessions
- **Model Dynamics**: "Consensus forming…" when multiple models agree
- **Wellbeing Nudges**: "You've been thinking hard. Time for a break?"

13.2 Personality Modes

Choose your preferred personality style in **Settings → Delight**:

Mode	Description
Professional	Minimal, business-appropriate feedback
Subtle	Light touches of personality
Expressive	Full personality with humor
Playful	Maximum fun, includes easter eggs

13.3 Achievements

Earn achievements as you use Think Tank:

Achievement	How to Unlock
Domain Explorer	Explore 10+ knowledge domains
Week Warrior	Use Think Tank 7 days in a row
Renaissance Mind	Explore 50+ domains
Monthly Mind	30-day streak

View your achievements in **Settings → Achievements**.

13.4 Easter Eggs

Think Tank has hidden surprises! Try: - Typing special phrases - Using keyboard shortcuts - Exploring during special times

Discover them yourself—that's half the fun!

13.5 Sound Effects

Enable audio feedback in **Settings → Delight → Sounds**:

Theme	Style
Default	Pleasant chimes
Mission Control	NASA-inspired beeps
Library	Page turns, book sounds
Workshop	Tool clicks
Emissions	Tesla-style… fun

13.6 Customizing Delight

Toggle individual features: - Domain messages - Model personality - Time awareness - Achievements - Wellbeing nudges - Easter eggs - Sound effects

Set intensity level (1-10) to control how often messages appear.

Thank you for using Think Tank! We're constantly improving based on your feedback.

PART 4: TECHNICAL SECTIONS

SECTION-00-SHARED-TYPES-AND-CONSTANTS

SECTION 0: UNIFIED SHARED TYPES & CONSTANTS (v2.0.0 - SINGLE SOURCE OF TRUTH)

â•

> **CRITICAL: This section defines ALL shared types, interfaces, and constants. All other sections MUST import from `@radiant/shared` - NEVER redefine these.**

0.1 PROJECT ROOT CONFIGURATION

Directory Structure

```
radiant/
â"œâ"€â"€ package.json
â"œâ"€â"€ pnpm-workspace.yaml
â"œâ"€â"€ tsconfig.base.json
â"œâ"€â"€ .nvmrc
â"œâ"€â"€ .gitignore
â",
â"œâ"€â"€ packages/
â",   â"œâ"€â"€ shared/                    # SECTION 0 - Shared types (implement first)
â",   â""â"€â"€ infrastructure/            # SECTIONS 2-3 - CDK stacks
â",
â"œâ"€â"€ functions/                    # SECTIONS 4-5 - Lambda functions
â",
â"œâ"€â"€ migrations/                   # SECTION 7 - Database migrations
```

```
â",
â"œâ"€â"€ apps/
â",   â"œâ"€â"€ swift-deployer/           # SECTION 1 - macOS deployment app
â",   â""â"€â"€ admin-dashboard/          # SECTION 8 - Next.js admin UI
â",
â""â"€â"€ docs/
```

package.json (root)

```
{
  "name": "radiant",
  "version": "2.2.0",
  "private": true,
  "workspaces": [
    "packages/*",
    "apps/*",
    "functions/*"
  ],
  "scripts": {
    "build": "pnpm -r build",
    "build:shared": "cd packages/shared && pnpm build",
    "test": "pnpm -r test",
    "lint": "pnpm -r lint",
    "deploy:dev": "cd packages/infrastructure && cdk deploy --all --context environment=dev'
    "deploy:staging": "cd packages/infrastructure && cdk deploy --all --context environment=
    "deploy:prod": "cd packages/infrastructure && cdk deploy --all --context environment=pro
  },
  "devDependencies": {
    "@types/node": "^20.10.0",
    "typescript": "^5.3.0"
  },
  "engines": {
    "node": ">=20.0.0",
    "pnpm": ">=8.0.0"
  }
}
```

pnpm-workspace.yaml

```
packages:
  - 'packages/*'
  - 'apps/*'
  - 'functions/*'
```

tsconfig.base.json

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "NodeNext",
    "moduleResolution": "NodeNext",
    "lib": ["ES2022"],
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "declaration": true,
    "declarationMap": true,
    "sourceMap": true,
    "resolveJsonModule": true,
    "paths": {
      "@radiant/shared": ["./packages/shared/src"],
      "@radiant/shared/*": ["./packages/shared/src/*"]
    }
  }
}
```

.nvmrc

```
20
```

.gitignore

```
node_modules/
dist/
.next/
*.log
.env*
!.env.example
.DS_Store
coverage/
cdk.out/
```

0.2 SHARED PACKAGE STRUCTURE

```
packages/shared/
â"œâ"€â"€ package.json
â"œâ"€â"€ tsconfig.json
â""â"€â"€ src/
```

```
â"œâ"€â"€ index.ts
â"œâ"€â"€ types/
â",   â"œâ"€â"€ index.ts
â",   â"œâ"€â"€ app.types.ts
â",   â"œâ"€â"€ environment.types.ts
â",   â"œâ"€â"€ ai.types.ts
â",   â"œâ"€â"€ admin.types.ts
â",   â"œâ"€â"€ billing.types.ts
â",   â""â"€â"€ compliance.types.ts
â"œâ"€â"€ constants/
â",   â"œâ"€â"€ index.ts
â",   â"œâ"€â"€ apps.ts
â",   â"œâ"€â"€ environments.ts
â",   â"œâ"€â"€ tiers.ts
â",   â"œâ"€â"€ regions.ts
â",   â""â"€â"€ providers.ts
â""â"€â"€ utils/
    â"œâ"€â"€ index.ts
    â"œâ"€â"€ validation.ts
    â""â"€â"€ formatting.ts
```

packages/shared/package.json

```
{
  "name": "@radiant/shared",
  "version": "2.2.0",
  "main": "./dist/index.js",
  "types": "./dist/index.d.ts",
  "exports": {
    ".": "./dist/index.js",
    "./types": "./dist/types/index.js",
    "./constants": "./dist/constants/index.js",
    "./utils": "./dist/utils/index.js"
  },
  "scripts": {
    "build": "tsc",
    "clean": "rm -rf dist",
    "prepublishOnly": "pnpm build"
  },
  "devDependencies": {
    "typescript": "^5.3.0"
  }
}
```

packages/shared/tsconfig.json

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "outDir": "./dist",
    "rootDir": "./src"
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist"]
}
```

0.3 VERSION & CONSTANTS

packages/shared/src/constants/version.ts

```
/**
 * RADIANT Version Constants
 * SINGLE SOURCE OF TRUTH for version numbers
 *
 * @description Update this file when releasing new versions.
 * All other code should import from here.
 */

// ============================================================================
// VERSION INFORMATION
// ============================================================================

/** Current RADIANT platform version */
export const RADIANT_VERSION = '4.17.0';

/** Version components for programmatic comparison */
export const VERSION = {
  major: 4,
  minor: 17,
  patch: 0,
  full: '4.17.0',
  build: process.env.BUILD_NUMBER || 'local',
  date: '2024-12',
} as const;

/** Minimum supported versions for various components */
export const MIN_VERSIONS = {
  node: '20.0.0',
  npm: '10.0.0',
```

```
  cdk: '2.120.0',
  postgres: '15.0',
  swift: '5.9',
  macos: '13.0',
  xcode: '15.0',
} as const;

// ============================================================================
// DOMAIN CONFIGURATION
// ============================================================================

/**
 * Domain placeholder - replace with your actual domain
 * Used throughout the codebase for consistency
 */
export const DOMAIN_PLACEHOLDER = 'YOUR_DOMAIN.com';

/**
 * Check if domain has been configured
 */
export function isDomainConfigured(domain: string): boolean {
  return !domain.includes(DOMAIN_PLACEHOLDER);
}
```

packages/shared/src/constants/index.ts

```
// Re-export all constants
export * from './version';
```

0.4 TYPE DEFINITIONS

packages/shared/src/types/index.ts

```
// Re-export all types and constants
export * from './app.types';
export * from './environment.types';
export * from './ai.types';
export * from './admin.types';
export * from './billing.types';
export * from './compliance.types';

// Re-export constants
export * from '../constants';
```

packages/shared/src/types/app.types.ts

```typescript
/**
 * RADIANT v2.2.0 - Application Types
 * SINGLE SOURCE OF TRUTH
 */

import type { Environment, TierLevel } from './environment.types';

export interface ManagedApp {
  id: string;                    // Lowercase, no spaces: "thinktank"
  name: string;                  // Display name: "Think Tank"
  domain: string;                // Base domain: "app.YOUR_DOMAIN.com"
  description?: string;
  icon?: string;
  version?: string;
  status: AppStatus;
  environments: EnvironmentStatus[];
  createdAt: Date;
  updatedAt: Date;
}

export type AppStatus = 'active' | 'inactive' | 'maintenance' | 'deprecated';

export interface EnvironmentStatus {
  environment: Environment;
  status: DeploymentStatus;
  lastDeployed?: Date;
  version?: string;
  tier: TierLevel;
  region: string;
  endpoints?: EnvironmentEndpoints;
}

export type DeploymentStatus =
  | 'not_deployed'
  | 'deploying'
  | 'deployed'
  | 'failed'
  | 'updating'
  | 'destroying';

export interface EnvironmentEndpoints {
  api?: string;
  graphql?: string;
  admin?: string;
```

```
  dashboard?: string;
}

export interface AppConfig {
  appId: string;
  appName: string;
  domain: string;
  environments: Record<Environment, EnvironmentConfig>;
}

export interface EnvironmentConfig {
  tier: TierLevel;
  region: string;
  enabledFeatures: FeatureFlags;
  customConfig?: Record<string, unknown>;
}

export interface FeatureFlags {
  selfHostedModels: boolean;
  multiRegion: boolean;
  waf: boolean;
  guardDuty: boolean;
  hipaaCompliance: boolean;
  advancedAnalytics: boolean;
  customBranding: boolean;
  sla: boolean;
}

export type DeploymentPhase =
  | 'idle'
  | 'initializing'
  | 'validating'
  | 'bootstrap'
  | 'foundation'
  | 'networking'
  | 'security'
  | 'data'
  | 'storage'
  | 'auth'
  | 'ai'
  | 'api'
  | 'admin'
  | 'migrations'
  | 'verification'
  | 'complete'
  | 'failed';
```

```typescript
export interface DeploymentProgress {
  phase: DeploymentPhase;
  progress: number;              // 0-100
  message: string;
  startedAt: Date;
  estimatedCompletion?: Date;
  logs: LogEntry[];
}

export interface LogEntry {
  timestamp: Date;
  level: 'debug' | 'info' | 'warn' | 'error';
  message: string;
  details?: Record<string, unknown>;
}
```

packages/shared/src/types/environment.types.ts

```typescript
/**
 * RADIANT v2.2.0 - Environment & Tier Types
 * SINGLE SOURCE OF TRUTH
 */

export type Environment = 'dev' | 'staging' | 'prod';

export interface EnvironmentInfo {
  name: Environment;
  displayName: string;
  color: string;
  requiresApproval: boolean;
  minTier: TierLevel;
  defaultTier: TierLevel;
}

export type TierLevel = 1 | 2 | 3 | 4 | 5;

export type TierName = 'SEED' | 'STARTUP' | 'GROWTH' | 'SCALE' | 'ENTERPRISE';

export interface TierConfig {
  level: TierLevel;
  name: TierName;
  description: string;

  // Compute
  vpcCidr: string;
```

```
  azCount: number;
  natGateways: number;

  // Database
  auroraMinCapacity: number;
  auroraMaxCapacity: number;
  enableGlobalDatabase: boolean;

  // Cache
  elasticacheNodes: number;
  elasticacheNodeType: string;

  // AI
  enableSelfHostedModels: boolean;
  maxSagemakerEndpoints: number;
  litellmTaskCount: number;
  litellmCpu: number;
  litellmMemory: number;

  // Security
  enableWaf: boolean;
  enableGuardDuty: boolean;
  enableSecurityHub: boolean;

  // Compliance
  enableHipaa: boolean;

  // Costs
  estimatedMonthlyCost: CostEstimate;
}

export interface CostEstimate {
  min: number;
  max: number;
  typical: number;
}

export interface RegionConfig {
  code: string;
  name: string;
  available: boolean;
  isGlobal: boolean;
}
```

packages/shared/src/types/ai.types.ts

```typescript
/**
 * RADIANT v2.2.0 - AI/Model Types
 * SINGLE SOURCE OF TRUTH
 */

export interface AIProvider {
  id: string;
  name: string;
  apiKeyEnvVar: string;
  baseUrl: string;
  hipaaCompliant: boolean;
  capabilities: ProviderCapability[];
  status: ProviderStatus;
  models?: AIModel[];
}

export type ProviderCapability =
  | 'text_generation'
  | 'chat_completion'
  | 'embeddings'
  | 'image_generation'
  | 'image_analysis'
  | 'video_generation'
  | 'audio_transcription'
  | 'audio_generation'
  | 'code_generation'
  | 'reasoning'
  | 'search'
  | 'function_calling'
  | '3d_generation';

export type ProviderStatus = 'active' | 'degraded' | 'maintenance' | 'disabled';

export interface AIModel {
  id: string;
  providerId: string;
  name: string;
  displayName: string;
  description?: string;
  specialty: ModelSpecialty;
  category: ModelCategory;
  capabilities: ProviderCapability[];
  contextWindow: number;
  maxOutputTokens: number;
```

```
  pricing: ModelPricing;
  status: ModelStatus;
  isHosted: boolean;
  thermalState?: ThermalState;
  sagemakerEndpoint?: string;
  metadata?: Record<string, unknown>;
}

export type ModelSpecialty =
  | 'text_generation'
  | 'reasoning'
  | 'coding'
  | 'image_generation'
  | 'image_analysis'
  | 'video_generation'
  | 'audio_transcription'
  | 'audio_generation'
  | 'embeddings'
  | 'search'
  | '3d_generation'
  | 'scientific';

export type ModelCategory =
  | 'llm'
  | 'vision'
  | 'audio'
  | 'multimodal'
  | 'embedding'
  | 'specialized';

export type ModelStatus = 'active' | 'disabled' | 'deprecated' | 'coming_soon';

export interface ModelPricing {
  inputPricePerMillion: number;
  outputPricePerMillion: number;
  imagePrice?: number;
  audioMinutePrice?: number;
  currency: 'USD';
}

// Thermal state for self-hosted models
export type ThermalState = 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';

export interface ThermalConfig {
  modelId: string;
  state: ThermalState;
```

```
  targetState?: ThermalState;
  lastStateChange?: Date;
  coldStartTime: number;
  warmupTime: number;
  idleTimeout: number;
  minInstances: number;
  maxInstances: number;
}

// Service state for mid-level services
export type ServiceState = 'RUNNING' | 'DEGRADED' | 'DISABLED' | 'OFFLINE';

export interface MidLevelService {
  id: string;
  name: string;
  description: string;
  state: ServiceState;
  dependencies: string[];
  healthEndpoint: string;
  lastHealthCheck?: Date;
  metrics?: ServiceMetrics;
}

export interface ServiceMetrics {
  requestsPerMinute: number;
  averageLatencyMs: number;
  errorRate: number;
  lastUpdated: Date;
}
```

packages/shared/src/types/admin.types.ts

```
/**
 * RADIANT v2.2.0 - Administrator Types
 * SINGLE SOURCE OF TRUTH
 */

import type { Environment } from './environment.types';

export interface Administrator {
  id: string;
  cognitoUserId: string;
  email: string;
  firstName: string;
  lastName: string;
  displayName: string;
```

```
  role: AdminRole;
  appId: string;
  tenantId: string;
  mfaEnabled: boolean;
  mfaMethod?: 'totp' | 'sms';
  status: AdminStatus;
  profile: AdminProfile;
  createdAt: Date;
  updatedAt: Date;
  createdBy?: string;
  lastLoginAt?: Date;
}

export type AdminRole = 'super_admin' | 'admin' | 'operator' | 'auditor';

export type AdminStatus = 'active' | 'inactive' | 'suspended' | 'pending';

export interface AdminRolePermissions {
  canManageAdmins: boolean;
  canManageModels: boolean;
  canManageProviders: boolean;
  canManageBilling: boolean;
  canDeploy: boolean;
  canApprove: boolean;
  canViewAuditLogs: boolean;
}

export const ROLE_PERMISSIONS: Record<AdminRole, AdminRolePermissions> = {
  super_admin: {
    canManageAdmins: true,
    canManageModels: true,
    canManageProviders: true,
    canManageBilling: true,
    canDeploy: true,
    canApprove: true,
    canViewAuditLogs: true,
  },
  admin: {
    canManageAdmins: true,
    canManageModels: true,
    canManageProviders: true,
    canManageBilling: true,
    canDeploy: true,
    canApprove: true,
    canViewAuditLogs: true,
  },
```

```
  operator: {
    canManageAdmins: false,
    canManageModels: true,
    canManageProviders: true,
    canManageBilling: false,
    canDeploy: true,
    canApprove: false,
    canViewAuditLogs: false,
  },
  auditor: {
    canManageAdmins: false,
    canManageModels: false,
    canManageProviders: false,
    canManageBilling: false,
    canDeploy: false,
    canApprove: false,
    canViewAuditLogs: true,
  },
};

export interface AdminProfile {
  timezone: string;
  language: string;
  dateFormat: string;
  timeFormat: string;
  currency: string;
  notifications: NotificationPreferences;
  ui: UIPreferences;
}

export interface NotificationPreferences {
  emailAlerts: boolean;
  slackAlerts: boolean;
  smsAlerts: boolean;
  alertTypes: string[];
}

export interface UIPreferences {
  theme: 'light' | 'dark' | 'system';
  sidebarCollapsed: boolean;
  defaultEnvironment: Environment;
  tableRowsPerPage: number;
}

export interface Invitation {
  id: string;
```

```
  email: string;
  role: AdminRole;
  invitedBy: string;
  appId: string;
  tenantId: string;
  environment?: Environment;
  tokenHash: string;
  expiresAt: Date;
  status: InvitationStatus;
  message?: string;
  createdAt: Date;
  acceptedAt?: Date;
}

export type InvitationStatus = 'pending' | 'accepted' | 'expired' | 'revoked';

export interface ApprovalRequest {
  id: string;
  type: ApprovalType;
  action: string;
  targetId: string;
  targetType: string;
  requestedBy: string;
  appId: string;
  tenantId: string;
  environment: Environment;
  status: ApprovalStatus;
  details: Record<string, unknown>;
  requiredApprovers: number;
  approvals: ApprovalVote[];
  expiresAt: Date;
  createdAt: Date;
  completedAt?: Date;
}

export type ApprovalType = 'deployment' | 'promotion' | 'config_change' | 'admin_action';

export type ApprovalStatus = 'pending' | 'approved' | 'rejected' | 'expired';

export interface ApprovalVote {
  adminId: string;
  vote: 'approve' | 'reject';
  comment?: string;
  votedAt: Date;
}
```

packages/shared/src/types/billing.types.ts

```typescript
/**
 * RADIANT v2.2.0 - Billing/Metering Types
 * SINGLE SOURCE OF TRUTH
 */

export interface UsageEvent {
  id: string;
  tenantId: string;
  appId: string;
  userId?: string;
  modelId: string;
  providerId: string;
  inputTokens: number;
  outputTokens: number;
  totalTokens: number;
  inputCost: number;
  outputCost: number;
  totalCost: number;
  margin: number;
  billedAmount: number;
  requestType: RequestType;
  responseTime: number;
  status: UsageStatus;
  timestamp: Date;
  metadata?: Record<string, unknown>;
}

export type RequestType = 'chat' | 'completion' | 'embedding' | 'image' | 'audio' | 'video';

export type UsageStatus = 'success' | 'error' | 'rate_limited' | 'timeout';

export interface TenantBilling {
  tenantId: string;
  appId: string;
  stripeCustomerId?: string;
  billingEmail: string;
  plan: BillingPlan;
  margin: number;
  creditBalance: number;
  lastInvoiceDate?: Date;
  nextInvoiceDate?: Date;
}

export type BillingPlan = 'free' | 'starter' | 'professional' | 'enterprise';
```

```typescript
export interface Invoice {
  id: string;
  tenantId: string;
  appId: string;
  stripeInvoiceId?: string;
  periodStart: Date;
  periodEnd: Date;
  subtotal: number;
  margin: number;
  tax: number;
  total: number;
  status: InvoiceStatus;
  paidAt?: Date;
  createdAt: Date;
}

export type InvoiceStatus = 'draft' | 'pending' | 'paid' | 'failed' | 'void';

export interface BillingBreakdown {
  period: { start: Date; end: Date };
  usage: {
    totalRequests: number;
    totalTokens: number;
    byModel: ModelUsageSummary[];
    byProvider: ProviderUsageSummary[];
  };
  costs: {
    baseCost: number;
    margin: number;
    total: number;
  };
}

export interface ModelUsageSummary {
  modelId: string;
  modelName: string;
  requests: number;
  tokens: number;
  cost: number;
}

export interface ProviderUsageSummary {
  providerId: string;
  providerName: string;
  requests: number;
```

```
  tokens: number;
  cost: number;
}

export interface PricingConfig {
  defaultMargin: number;
  minimumCharge: number;
  currencyCode: string;
  tierDiscounts: TierDiscount[];
}

export interface TierDiscount {
  minTokens: number;
  discountPercent: number;
}

export const DEFAULT_PRICING: PricingConfig = {
  defaultMargin: 0.40,
  minimumCharge: 0.01,
  currencyCode: 'USD',
  tierDiscounts: [
    { minTokens: 1_000_000, discountPercent: 5 },
    { minTokens: 10_000_000, discountPercent: 10 },
    { minTokens: 100_000_000, discountPercent: 15 },
  ],
};
```

packages/shared/src/types/compliance.types.ts

```
/**
 * RADIANT v2.2.0 - Compliance/PHI Types
 * SINGLE SOURCE OF TRUTH
 */

export type PHIMode = 'auto' | 'manual' | 'disabled';

export interface PHIConfig {
  mode: PHIMode;
  categories: PHICategory[];
  autoSanitize: boolean;
  allowReidentification: boolean;
  logSanitization: boolean;
  retentionDays: number;
}

export type PHICategory =
```

```
  | 'name'
  | 'date'
  | 'phone'
  | 'email'
  | 'ssn'
  | 'mrn'
  | 'address'
  | 'age'
  | 'medical_condition'
  | 'medication'
  | 'procedure';

export const DEFAULT_PHI_CONFIG: PHIConfig = {
  mode: 'auto',
  categories: ['name', 'date', 'phone', 'email', 'ssn', 'mrn', 'address'],
  autoSanitize: true,
  allowReidentification: false,
  logSanitization: true,
  retentionDays: 365,
};

export interface ComplianceReport {
  id: string;
  type: ComplianceReportType;
  generatedAt: Date;
  period: { start: Date; end: Date };
  results: ComplianceCheck[];
  overallStatus: ComplianceStatus;
  recommendations: string[];
}

export type ComplianceReportType = 'hipaa' | 'soc2' | 'gdpr' | 'full';

export type ComplianceStatus = 'compliant' | 'non_compliant' | 'needs_review';

export interface ComplianceCheck {
  id: string;
  name: string;
  description: string;
  category: string;
  status: ComplianceStatus;
  evidence?: string;
  remediation?: string;
}

export interface AuditLog {
```

```
  id: string;
  tenantId: string;
  appId: string;
  adminId?: string;
  action: string;
  resource: string;
  resourceId: string;
  details: Record<string, unknown>;
  ipAddress?: string;
  userAgent?: string;
  timestamp: Date;
}
```

0.4 CONSTANTS

packages/shared/src/constants/index.ts

```
export * from './apps';
export * from './environments';
export * from './tiers';
export * from './regions';
export * from './providers';
```

packages/shared/src/constants/apps.ts

```
/**
 * RADIANT v2.2.0 - Managed Applications
 * SINGLE SOURCE OF TRUTH - Update YOUR_DOMAIN.com with your actual domain
 */

export const MANAGED_APPS = [
  { id: 'thinktank', name: 'Think Tank', domain: 'thinktank.YOUR_DOMAIN.com' },
  { id: 'launchboard', name: 'Launch Board', domain: 'launchboard.YOUR_DOMAIN.com' },
  { id: 'alwaysme', name: 'Always Me', domain: 'alwaysme.YOUR_DOMAIN.com' },
  { id: 'mechanicalmaker', name: 'Mechanical Maker', domain: 'mechanicalmaker.YOUR_DOMAIN.co
] as const;

export type ManagedAppId = typeof MANAGED_APPS[number]['id'];
```

packages/shared/src/constants/environments.ts

```
/**
 * RADIANT v2.2.0 - Environment Configuration
 * SINGLE SOURCE OF TRUTH
 */
```

```typescript
import type { Environment, EnvironmentInfo, TierLevel } from '../types';

export const ENVIRONMENTS: Record<Environment, EnvironmentInfo> = {
  dev: {
    name: 'dev',
    displayName: 'Development',
    color: '#3B82F6',
    requiresApproval: false,
    minTier: 1 as TierLevel,
    defaultTier: 1 as TierLevel,
  },
  staging: {
    name: 'staging',
    displayName: 'Staging',
    color: '#F59E0B',
    requiresApproval: false,
    minTier: 2 as TierLevel,
    defaultTier: 2 as TierLevel,
  },
  prod: {
    name: 'prod',
    displayName: 'Production',
    color: '#EF4444',
    requiresApproval: true,
    minTier: 3 as TierLevel,
    defaultTier: 3 as TierLevel,
  },
};

export const ENVIRONMENT_LIST: Environment[] = ['dev', 'staging', 'prod'];
```

packages/shared/src/constants/tiers.ts

```typescript
/**
 * RADIANT v2.2.0 - Infrastructure Tiers
 * SINGLE SOURCE OF TRUTH - Used by CDK and all components
 */

import type { TierConfig, TierLevel, TierName } from '../types';

export const TIER_NAMES: Record<TierLevel, TierName> = {
  1: 'SEED',
  2: 'STARTUP',
  3: 'GROWTH',
  4: 'SCALE',
```

```
  5: 'ENTERPRISE',
};

export const TIER_CONFIGS: Record<TierLevel, TierConfig> = {
  1: {
    level: 1,
    name: 'SEED',
    description: 'Development and testing, minimal costs',
    vpcCidr: '10.0.0.0/20',
    azCount: 2,
    natGateways: 1,
    auroraMinCapacity: 0.5,
    auroraMaxCapacity: 2,
    enableGlobalDatabase: false,
    elasticacheNodes: 0,
    elasticacheNodeType: 'cache.t4g.micro',
    enableSelfHostedModels: false,
    maxSagemakerEndpoints: 0,
    litellmTaskCount: 1,
    litellmCpu: 256,
    litellmMemory: 512,
    enableWaf: false,
    enableGuardDuty: false,
    enableSecurityHub: false,
    enableHipaa: false,
    estimatedMonthlyCost: { min: 50, max: 150, typical: 85 },
  },
  2: {
    level: 2,
    name: 'STARTUP',
    description: 'Small production workloads',
    vpcCidr: '10.0.0.0/18',
    azCount: 2,
    natGateways: 1,
    auroraMinCapacity: 1,
    auroraMaxCapacity: 8,
    enableGlobalDatabase: false,
    elasticacheNodes: 1,
    elasticacheNodeType: 'cache.t4g.small',
    enableSelfHostedModels: false,
    maxSagemakerEndpoints: 0,
    litellmTaskCount: 2,
    litellmCpu: 512,
    litellmMemory: 1024,
    enableWaf: true,
    enableGuardDuty: true,
```

```
  enableSecurityHub: false,
  enableHipaa: false,
  estimatedMonthlyCost: { min: 200, max: 400, typical: 255 },
},
3: {
  level: 3,
  name: 'GROWTH',
  description: 'Medium production with self-hosted models',
  vpcCidr: '10.0.0.0/17',
  azCount: 3,
  natGateways: 2,
  auroraMinCapacity: 2,
  auroraMaxCapacity: 16,
  enableGlobalDatabase: false,
  elasticacheNodes: 2,
  elasticacheNodeType: 'cache.r6g.large',
  enableSelfHostedModels: true,
  maxSagemakerEndpoints: 10,
  litellmTaskCount: 3,
  litellmCpu: 1024,
  litellmMemory: 2048,
  enableWaf: true,
  enableGuardDuty: true,
  enableSecurityHub: true,
  enableHipaa: true,
  estimatedMonthlyCost: { min: 1000, max: 2500, typical: 1475 },
},
4: {
  level: 4,
  name: 'SCALE',
  description: 'Large production with multi-region',
  vpcCidr: '10.0.0.0/16',
  azCount: 3,
  natGateways: 3,
  auroraMinCapacity: 4,
  auroraMaxCapacity: 64,
  enableGlobalDatabase: true,
  elasticacheNodes: 3,
  elasticacheNodeType: 'cache.r6g.xlarge',
  enableSelfHostedModels: true,
  maxSagemakerEndpoints: 30,
  litellmTaskCount: 5,
  litellmCpu: 2048,
  litellmMemory: 4096,
  enableWaf: true,
  enableGuardDuty: true,
```

```
    enableSecurityHub: true,
    enableHipaa: true,
    estimatedMonthlyCost: { min: 4000, max: 8000, typical: 5450 },
  },
  5: {
    level: 5,
    name: 'ENTERPRISE',
    description: 'Enterprise-grade global deployment',
    vpcCidr: '10.0.0.0/14',
    azCount: 3,
    natGateways: 3,
    auroraMinCapacity: 8,
    auroraMaxCapacity: 128,
    enableGlobalDatabase: true,
    elasticacheNodes: 6,
    elasticacheNodeType: 'cache.r6g.2xlarge',
    enableSelfHostedModels: true,
    maxSagemakerEndpoints: 100,
    litellmTaskCount: 10,
    litellmCpu: 4096,
    litellmMemory: 8192,
    enableWaf: true,
    enableGuardDuty: true,
    enableSecurityHub: true,
    enableHipaa: true,
    estimatedMonthlyCost: { min: 15000, max: 35000, typical: 21500 },
  },
};

export function getTierConfig(tier: TierLevel): TierConfig {
  return TIER_CONFIGS[tier];
}

export function getTierName(tier: TierLevel): TierName {
  return TIER_NAMES[tier];
}

export function validateTierForEnvironment(tier: TierLevel, environment: string): void {
  if (environment === 'prod' && tier < 3) {
    throw new Error('Production requires Tier 3 (GROWTH) or higher');
  }
  if (environment === 'staging' && tier < 2) {
    throw new Error('Staging requires Tier 2 (STARTUP) or higher');
  }
}
```

```typescript
export function getFeatureFlagsForTier(tier: TierLevel) {
  const config = TIER_CONFIGS[tier];
  return {
    selfHostedModels: config.enableSelfHostedModels,
    multiRegion: config.enableGlobalDatabase,
    waf: config.enableWaf,
    guardDuty: config.enableGuardDuty,
    hipaaCompliance: config.enableHipaa,
    advancedAnalytics: tier >= 3,
    customBranding: tier >= 4,
    sla: tier >= 4,
  };
}
```

packages/shared/src/constants/regions.ts

```typescript
/**
 * RADIANT v2.2.0 - AWS Region Configuration
 * SINGLE SOURCE OF TRUTH
 */

import type { RegionConfig } from '../types';

export const REGIONS: Record<string, RegionConfig> = {
  'us-east-1': { code: 'us-east-1', name: 'US East (N. Virginia)', available: true, isGlobal
  'us-west-2': { code: 'us-west-2', name: 'US West (Oregon)', available: true, isGlobal: fal
  'eu-west-1': { code: 'eu-west-1', name: 'Europe (Ireland)', available: true, isGlobal: tru
  'eu-central-1': { code: 'eu-central-1', name: 'Europe (Frankfurt)', available: true, isGlo
  'ap-northeast-1': { code: 'ap-northeast-1', name: 'Asia Pacific (Tokyo)', available: true,
  'ap-southeast-1': { code: 'ap-southeast-1', name: 'Asia Pacific (Singapore)', available: t
  'ap-south-1': { code: 'ap-south-1', name: 'Asia Pacific (Mumbai)', available: true, isGlob
};

export const PRIMARY_REGION = 'us-east-1';

export const MULTI_REGION_CONFIG = {
  primary: 'us-east-1',
  europe: 'eu-west-1',
  asia: 'ap-northeast-1',
} as const;

export function getMultiRegionDeployment(primaryRegion: string): string[] {
  if (primaryRegion === 'us-east-1') {
    return ['us-east-1', 'eu-west-1', 'ap-northeast-1'];
  }
  if (primaryRegion.startsWith('eu-')) {
```

```
    return ['eu-west-1', 'us-east-1', 'ap-northeast-1'];
  }
  if (primaryRegion.startsWith('ap-')) {
    return ['ap-northeast-1', 'us-east-1', 'eu-west-1'];
  }
  return [primaryRegion];
}

export function getRegionConfig(region: string): RegionConfig {
  const config = REGIONS[region];
  if (!config) {
    throw new Error(\`Unknown region: \${region}\`);
  }
  return config;
}

export function isValidRegion(region: string): boolean {
  return region in REGIONS;
}
```

packages/shared/src/constants/providers.ts

```
/**
 * RADIANT v2.2.0 - External AI Providers
 * SINGLE SOURCE OF TRUTH
 */

import type { ProviderCapability } from '../types';

export interface ExternalProviderInfo {
  id: string;
  name: string;
  hipaaCompliant: boolean;
  capabilities: ProviderCapability[];
}

export const EXTERNAL_PROVIDERS: ExternalProviderInfo[] = [
  { id: 'openai', name: 'OpenAI', hipaaCompliant: true, capabilities: ['text_generation', 'c
  { id: 'anthropic', name: 'Anthropic', hipaaCompliant: true, capabilities: ['text_generatio
  { id: 'google', name: 'Google AI', hipaaCompliant: true, capabilities: ['text_generation',
  { id: 'xai', name: 'xAI', hipaaCompliant: false, capabilities: ['text_generation', 'chat_c
  { id: 'perplexity', name: 'Perplexity', hipaaCompliant: false, capabilities: ['text_genera
  { id: 'deepseek', name: 'DeepSeek', hipaaCompliant: false, capabilities: ['text_generation
  { id: 'mistral', name: 'Mistral AI', hipaaCompliant: false, capabilities: ['text_generatio
  { id: 'cohere', name: 'Cohere', hipaaCompliant: true, capabilities: ['text_generation', 'c
  { id: 'together', name: 'Together AI', hipaaCompliant: false, capabilities: ['text_generat
```

```
  { id: 'groq', name: 'Groq', hipaaCompliant: false, capabilities: ['text_generation', 'chat
  { id: 'fireworks', name: 'Fireworks AI', hipaaCompliant: false, capabilities: ['text_gener
  { id: 'replicate', name: 'Replicate', hipaaCompliant: false, capabilities: ['image_generat
  { id: 'huggingface', name: 'Hugging Face', hipaaCompliant: false, capabilities: ['text_ger
  { id: 'anyscale', name: 'Anyscale', hipaaCompliant: false, capabilities: ['text_generation
  { id: 'databricks', name: 'Databricks', hipaaCompliant: true, capabilities: ['text_generat
  { id: 'aws_bedrock', name: 'AWS Bedrock', hipaaCompliant: true, capabilities: ['text_gener
  { id: 'azure_openai', name: 'Azure OpenAI', hipaaCompliant: true, capabilities: ['text_ger
  { id: 'vertex', name: 'Google Vertex AI', hipaaCompliant: true, capabilities: ['text_gener
  { id: 'ollama', name: 'Ollama (Local)', hipaaCompliant: true, capabilities: ['text_generat
  { id: 'elevenlabs', name: 'ElevenLabs', hipaaCompliant: false, capabilities: ['audio_gener
  { id: 'runway', name: 'Runway ML', hipaaCompliant: false, capabilities: ['video_generation
];

export const PROVIDER_ENDPOINTS: Record<string, string> = {
  openai: 'https://api.openai.com/v1',
  anthropic: 'https://api.anthropic.com/v1',
  google: 'https://generativelanguage.googleapis.com/v1beta',
  xai: 'https://api.x.ai/v1',
  perplexity: 'https://api.perplexity.ai',
  deepseek: 'https://api.deepseek.com/v1',
  mistral: 'https://api.mistral.ai/v1',
  cohere: 'https://api.cohere.ai/v1',
  together: 'https://api.together.xyz/v1',
  groq: 'https://api.groq.com/openai/v1',
  fireworks: 'https://api.fireworks.ai/inference/v1',
  replicate: 'https://api.replicate.com/v1',
  huggingface: 'https://api-inference.huggingface.co',
  elevenlabs: 'https://api.elevenlabs.io/v1',
  runway: 'https://api.runwayml.com/v1',
};

export function getProviderInfo(providerId: string): ExternalProviderInfo | undefined {
  return EXTERNAL_PROVIDERS.find(p => p.id === providerId);
}

export function getHipaaCompliantProviders(): ExternalProviderInfo[] {
  return EXTERNAL_PROVIDERS.filter(p => p.hipaaCompliant);
}
```

0.5 UTILITY FUNCTIONS

packages/shared/src/utils/index.ts

```
export * from './validation';
export * from './formatting';
```

packages/shared/src/utils/validation.ts

```
/**
 * RADIANT v2.2.0 - Validation Utilities
 */

export function isValidAppId(id: string): boolean {
  return /^[a-z][a-z0-9-]{2,30}$/.test(id);
}

export function isValidEmail(email: string): boolean {
  return /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);
}

export function isValidDomain(domain: string): boolean {
  return /^([a-z0-9-]+\.)+[a-z]{2,}$/i.test(domain);
}

export function isValidAWSRegion(region: string): boolean {
  return /^[a-z]{2}-[a-z]+-\d$/.test(region);
}

export function isValidAWSAccountId(accountId: string): boolean {
  return /^\d{12}$/.test(accountId);
}

export function sanitizeAppId(name: string): string {
  return name
    .toLowerCase()
    .replace(/[^a-z0-9-]/g, '-')
    .replace(/-+/g, '-')
    .replace(/^-|-$/g, '')
    .slice(0, 30);
}

export function validateRequired<T>(value: T | undefined | null, fieldName: string): T {
  if (value === undefined || value === null) {
    throw new Error(\`\${fieldName} is required\`);
  }
  return value;
```

```
}
```

packages/shared/src/utils/formatting.ts

```
/**
 * RADIANT v2.2.0 - Formatting Utilities
 */

export function formatCurrency(amount: number, currency = 'USD'): string {
  return new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency,
    minimumFractionDigits: 2,
    maximumFractionDigits: 4,
  }).format(amount);
}

export function formatNumber(num: number): string {
  return new Intl.NumberFormat('en-US').format(num);
}

export function formatTokens(tokens: number): string {
  if (tokens >= 1_000_000) {
    return \`\${(tokens / 1_000_000).toFixed(1)}M\`;
  }
  if (tokens >= 1_000) {
    return \`\${(tokens / 1_000).toFixed(1)}K\`;
  }
  return tokens.toString();
}

export function formatBytes(bytes: number): string {
  const units = ['B', 'KB', 'MB', 'GB', 'TB'];
  let unitIndex = 0;
  let value = bytes;

  while (value >= 1024 && unitIndex < units.length - 1) {
    value /= 1024;
    unitIndex++;
  }

  return \`\${value.toFixed(1)} \${units[unitIndex]}\`;
}

export function formatDuration(ms: number): string {
  if (ms < 1000) return \`\${ms}ms\`;
```

```
  if (ms < 60000) return \`\${(ms / 1000).toFixed(1)}s\`;
  if (ms < 3600000) return \`\${Math.floor(ms / 60000)}m \${Math.floor((ms % 60000) / 1000)}
  return \`\${Math.floor(ms / 3600000)}h \${Math.floor((ms % 3600000) / 60000)}m\`;
}

export function formatDate(date: Date): string {
  return new Intl.DateTimeFormat('en-US', {
    year: 'numeric',
    month: 'short',
    day: 'numeric',
  }).format(date);
}

export function formatDateTime(date: Date): string {
  return new Intl.DateTimeFormat('en-US', {
    year: 'numeric',
    month: 'short',
    day: 'numeric',
    hour: '2-digit',
    minute: '2-digit',
  }).format(date);
}
```

packages/shared/src/index.ts

```
/**
 * RADIANT v2.2.0 - Shared Package Main Export
 *
 * Usage in other packages:
 *   import { TierConfig, getTierConfig, formatCurrency } from '@radiant/shared';
 */

// Types
export * from './types';

// Constants
export * from './constants';

// Utils
export * from './utils';
```

0.6 BUILD AND VERIFY

After implementing Section 0, build and verify:

```
cd packages/shared
pnpm install
pnpm build

# Verify the build output
ls -la dist/
```

Expected output structure:

```
dist/
â"œâ"€â"€ index.js
â"œâ"€â"€ index.d.ts
â"œâ"€â"€ types/
â"¦   â"œâ"€â"€ index.js
â"¦   â"œâ"€â"€ index.d.ts
â"¦   â""â"€â"€ ...
â"œâ"€â"€ constants/
â"¦   â"œâ"€â"€ index.js
â"¦   â"œâ"€â"€ index.d.ts
â"¦   â""â"€â"€ ...
â""â"€â"€ utils/
    â"œâ"€â"€ index.js
    â"œâ"€â"€ index.d.ts
    â""â"€â"€ ...
```

â• â•

END OF SECTION 0

â• â•

â• â•

SECTION-01-SWIFT-DEPLOYMENT-APP

SECTION 1: FOUNDATION & SWIFT DEPLOYMENT APP (v2.0.0)

â• â•

> **Dependencies:** Section 0 (Shared Types) **Creates:** Monorepo structure, Swift macOS deployment app

1.1 OVERVIEW

This section creates: 1. **Monorepo foundation** - Already partially in Section 0 2. **Swift macOS Deployment App** - Complete GUI for deploying RADIANT

The Swift app is the primary interface for administrators. Users should NEVER need to open a terminal.

1.2 CDK INFRASTRUCTURE PACKAGE SETUP

Since types are now in @radiant/shared, the infrastructure package imports them:

packages/infrastructure/package.json

```
{
  "name": "@radiant/infrastructure",
  "version": "2.2.0",
  "private": true,
  "scripts": {
    "build": "tsc",
    "cdk": "cdk",
    "synth": "cdk synth",
    "deploy": "cdk deploy",
    "destroy": "cdk destroy"
  },
  "dependencies": {
    "@radiant/shared": "workspace:*",
    "aws-cdk-lib": "^2.120.0",
    "constructs": "^10.3.0",
    "source-map-support": "^0.5.21"
  },
  "devDependencies": {
    "@types/node": "^20.10.0",
    "aws-cdk": "^2.120.0",
    "typescript": "^5.3.0"
  }
}
```

packages/infrastructure/lib/config/tags.ts

```
/**
 * CDK-specific tagging utilities
 * Types imported from @radiant/shared
 */
```

```typescript
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';

export interface RadiantTags {
  project: string;
  environment: string;
  appId: string;
  tier: number;
  version?: string;
  costCenter?: string;
}

export function applyTags(scope: Construct, tags: RadiantTags): void {
  cdk.Tags.of(scope).add('Project', tags.project);
  cdk.Tags.of(scope).add('Environment', tags.environment);
  cdk.Tags.of(scope).add('AppId', tags.appId);
  cdk.Tags.of(scope).add('Tier', tags.tier.toString());
  cdk.Tags.of(scope).add('Version', tags.version || '2.2.0');
  cdk.Tags.of(scope).add('ManagedBy', 'CDK');

  if (tags.costCenter) {
    cdk.Tags.of(scope).add('CostCenter', tags.costCenter);
  }
}

export function getResourceName(
  appId: string,
  environment: string,
  resource: string,
  suffix?: string
): string {
  const base = \`\${appId}-\${environment}-\${resource}\`;
  return suffix ? \`\${base}-\${suffix}\` : base;
}
```

NOTE: The `tiers.ts` and `regions.ts` files are NOT created here.
CDK stacks import these directly from `@radiant/shared`:

```typescript
import { getTierConfig, TIER_CONFIGS, REGIONS } from '@radiant/shared';
```

1.3 SWIFT DEPLOYMENT APP

AI Implementation Notes

```
SWIFT FILE CREATION ORDER (for AI)
```

```
PHASE 1: Project Setup (create first)

1. Package.swift (or Xcode project)
2. Info.plist
3. RadiantDeployer.entitlements

PHASE 2: Core Files (no dependencies)

4. RadiantDeployerApp.swift (entry point)
5. Models/ManagedApp.swift
6. Models/Credentials.swift
7. Models/Deployment.swift

PHASE 3: State Management

8. AppState.swift (depends on: Models)

PHASE 4: Services (depend on Models)

9.  Services/CredentialService.swift
10. Services/CDKService.swift
11. Services/AWSService.swift
12. Services/APIService.swift

PHASE 5: Views (depend on AppState, Services)

13. Views/MainView.swift
14. Views/AppsView.swift
15. Views/DeployView.swift
16. Views/SettingsView.swift
17. Views/ProvidersView.swift
18. Views/ModelsView.swift
```

Platform Requirements

```
Platform: macOS 13.0+ (Ventura)
Swift: 5.9+
Xcode: 15.0+
Architecture: Universal (arm64 + x86_64)
```

RadiantDeployer/Package.swift

```
// swift-tools-version: 5.9
// RADIANT Deployer - macOS Swift Package
// Platform: macOS 13.0+

import PackageDescription

let package = Package(
    name: "RadiantDeployer",
    platforms: [
        .macOS(.v13)
    ],
    products: [
        .executable(name: "RadiantDeployer", targets: ["RadiantDeployer"])
    ],
    dependencies: [
        // SQLCipher for encrypted credential storage
        .package(url: "https://github.com/nicklockwood/GRDB.swift.git", from: "6.24.0"),
    ],
    targets: [
        .executableTarget(
            name: "RadiantDeployer",
            dependencies: [
                .product(name: "GRDB", package: "GRDB.swift"),
            ],
            path: "Sources",
            resources: [
                .copy("Resources/Infrastructure"),
                .copy("Resources/NodeRuntime"),
            ]
        ),
        .testTarget(
            name: "RadiantDeployerTests",
            dependencies: ["RadiantDeployer"],
            path: "Tests"
        ),
    ]
)
```

RadiantDeployer/Info.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList
<plist version="1.0">
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>en</string>
    <key>CFBundleExecutable</key>
    <string>RadiantDeployer</string>
    <key>CFBundleIconFile</key>
    <string>AppIcon</string>
    <key>CFBundleIdentifier</key>
    <string>com.radiant.deployer</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>RADIANT Deployer</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleShortVersionString</key>
    <string>4.17.0</string>
    <key>CFBundleVersion</key>
    <string>1</string>
    <key>LSMinimumSystemVersion</key>
    <string>13.0</string>
    <key>NSHumanReadableCopyright</key>
    <string>Copyright © 2024 RADIANT. All rights reserved.</string>
    <key>NSMainStoryboardFile</key>
    <string></string>
    <key>NSPrincipalClass</key>
    <string>NSApplication</string>
    <key>LSApplicationCategoryType</key>
    <string>public.app-category.developer-tools</string>
</dict>
</plist>
```

RadiantDeployer/RadiantDeployer.entitlements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList
<plist version="1.0">
<dict>
    <!-- App Sandbox (required for Mac App Store, optional otherwise) -->
    <key>com.apple.security.app-sandbox</key>
```

```
    <false/>

    <!-- Network access for AWS API calls -->
    <key>com.apple.security.network.client</key>
    <true/>

    <!-- File access for CDK operations -->
    <key>com.apple.security.files.user-selected.read-write</key>
    <true/>

    <!-- Keychain access for credential storage -->
    <key>com.apple.security.keychain</key>
    <true/>

    <!-- Process execution for CDK/Node commands -->
    <key>com.apple.security.cs.allow-unsigned-executable-memory</key>
    <true/>
</dict>
</plist>
```

RadiantDeployer/RadiantDeployerApp.swift

```swift
// MARK: - RADIANT Deployer App Entry Point
// Platform: macOS 13.0+
// Swift: 5.9+
// Dependencies: SwiftUI (system), AppState.swift

import SwiftUI

/// Main application entry point for RADIANT Deployer
/// This is the first file Xcode will look for when launching the app
@main
struct RadiantDeployerApp: App {
    // MARK: - State

    /// App-lifetime state object - created once, shared across all views
    @StateObject private var appState = AppState()

    // MARK: - Body

    var body: some Scene {
        WindowGroup {
            MainView()
                .environmentObject(appState)
```

```
                .frame(minWidth: 1200, minHeight: 800)
        }
        .windowStyle(.hiddenTitleBar)
        .commands {
            CommandGroup(replacing: .newItem) { }
        }

        Settings {
            SettingsView()
                .environmentObject(appState)
        }
    }
}

// MARK: - Preview

#Preview {
    MainView()
        .environmentObject(AppState())
        .frame(width: 1200, height: 800)
}
```

RadiantDeployer/AppState.swift

```
import SwiftUI
import Combine

@MainActor
final class AppState: ObservableObject {
    // MARK: - Navigation
    @Published var selectedTab: NavigationTab = .apps
    @Published var selectedApp: ManagedApp?
    @Published var selectedEnvironment: Environment = .dev

    // MARK: - Data
    @Published var apps: [ManagedApp] = []
    @Published var credentials: [CredentialSet] = []
    @Published var isLoading = false
    @Published var error: AppError?

    // MARK: - Deployment
    @Published var isDeploying = false
    @Published var deploymentProgress: DeploymentProgress?
    @Published var deploymentLogs: [LogEntry] = []

    // MARK: - Services
```

```
    let credentialService = CredentialService()
    let cdkService = CDKService()
    let awsService = AWSService()
    let apiService = APIService()

    // MARK: - Initialization
    init() {
        Task {
            await loadInitialData()
        }
    }

    func loadInitialData() async {
        isLoading = true
        defer { isLoading = false }

        do {
            credentials = try await credentialService.loadCredentials()
            apps = try await loadApps()
        } catch {
            self.error = AppError(message: "Failed to load data", underlying: error)
        }
    }

    private func loadApps() async throws -> [ManagedApp] {
        // Load from local storage or API
        return ManagedApp.defaults
    }
}

// MARK: - Navigation
enum NavigationTab: String, CaseIterable, Identifiable, Sendable {
    case apps = "Apps"
    case deploy = "Deploy"
    case providers = "Providers"
    case models = "Models"
    case settings = "Settings"

    var id: String { rawValue }

    var icon: String {
        switch self {
        case .apps: return "square.grid.2x2"
        case .deploy: return "arrow.up.circle"
        case .providers: return "building.2"
        case .models: return "cpu"
```

```
        case .settings: return "gearshape"
        }
    }
}

// MARK: - Environment
enum Environment: String, CaseIterable, Identifiable, Sendable {
    case dev = "Development"
    case staging = "Staging"
    case prod = "Production"

    var id: String { rawValue }

    var shortName: String {
        switch self {
        case .dev: return "DEV"
        case .staging: return "STAGING"
        case .prod: return "PROD"
        }
    }

    var color: Color {
        switch self {
        case .dev: return .blue
        case .staging: return .orange
        case .prod: return .green
        }
    }
}

// MARK: - Error
struct AppError: Identifiable, Sendable {
    let id = UUID()
    let message: String
    let underlying: (any Error)?

    var localizedDescription: String {
        if let underlying = underlying {
            return "\(message): \(underlying.localizedDescription)"
        }
        return message
    }
}
```

RadiantDeployer/Models/ManagedApp.swift

```swift
// MARK: - ManagedApp Model
// Platform: macOS 13.0+
// Dependencies: Foundation

import Foundation

// MARK: - Constants

/// Domain placeholder - replace with your actual domain during setup
let DOMAIN_PLACEHOLDER = "YOUR_DOMAIN.com"

// MARK: - ManagedApp

struct ManagedApp: Identifiable, Codable, Hashable, Sendable {
    let id: String
    var name: String
    var domain: String
    var description: String?
    var createdAt: Date
    var updatedAt: Date
    var environments: EnvironmentStatuses

    /// Check if domain has been configured
    var isDomainConfigured: Bool {
        !domain.contains(DOMAIN_PLACEHOLDER)
    }

    struct EnvironmentStatuses: Codable, Hashable, Sendable {
        var dev: EnvironmentStatus
        var staging: EnvironmentStatus
        var prod: EnvironmentStatus

        subscript(env: Environment) -> EnvironmentStatus {
            get {
                switch env {
                case .dev: return dev
                case .staging: return staging
                case .prod: return prod
                }
            }
            set {
                switch env {
                case .dev: dev = newValue
                case .staging: staging = newValue
```

```
                case .prod: prod = newValue
                }
            }
        }
    }
}

struct EnvironmentStatus: Codable, Hashable, Sendable {
    var deployed: Bool
    var version: String?
    var tier: Int
    var lastDeployedAt: Date?
    var healthStatus: HealthStatus
    var apiUrl: String?
    var dashboardUrl: String?
}

enum HealthStatus: String, Codable, Sendable {
    case healthy, degraded, unhealthy, unknown

    var color: String {
        switch self {
        case .healthy: return "green"
        case .degraded: return "orange"
        case .unhealthy: return "red"
        case .unknown: return "gray"
        }
    }
}

// MARK: - Defaults
extension ManagedApp {
    static let defaults: [ManagedApp] = [
        ManagedApp(
            id: "thinktank",
            name: "Think Tank",
            domain: "thinktank.\(DOMAIN_PLACEHOLDER)",
            description: "AI-powered brainstorming and ideation platform",
            createdAt: Date(),
            updatedAt: Date(),
            environments: .init(
                dev: .init(deployed: false, tier: 1, healthStatus: .unknown),
                staging: .init(deployed: false, tier: 2, healthStatus: .unknown),
                prod: .init(deployed: false, tier: 3, healthStatus: .unknown)
            )
        ),
```

```
            ManagedApp(
                id: "launchboard",
                name: "Launch Board",
                domain: "launchboard.\(DOMAIN_PLACEHOLDER)",
                description: "Project launch management and tracking",
                createdAt: Date(),
                updatedAt: Date(),
                environments: .init(
                    dev: .init(deployed: false, tier: 1, healthStatus: .unknown),
                    staging: .init(deployed: false, tier: 2, healthStatus: .unknown),
                    prod: .init(deployed: false, tier: 3, healthStatus: .unknown)
                )
            ),
            ManagedApp(
                id: "alwaysme",
                name: "Always Me",
                domain: "alwaysme.\(DOMAIN_PLACEHOLDER)",
                description: "Personal AI assistant and memory",
                createdAt: Date(),
                updatedAt: Date(),
                environments: .init(
                    dev: .init(deployed: false, tier: 1, healthStatus: .unknown),
                    staging: .init(deployed: false, tier: 2, healthStatus: .unknown),
                    prod: .init(deployed: false, tier: 3, healthStatus: .unknown)
                )
            ),
            ManagedApp(
                id: "mechanicalmaker",
                name: "Mechanical Maker",
                domain: "mechanicalmaker.\(DOMAIN_PLACEHOLDER)",
                description: "AI-assisted mechanical design and CAD",
                createdAt: Date(),
                updatedAt: Date(),
                environments: .init(
                    dev: .init(deployed: false, tier: 1, healthStatus: .unknown),
                    staging: .init(deployed: false, tier: 2, healthStatus: .unknown),
                    prod: .init(deployed: false, tier: 3, healthStatus: .unknown)
                )
            )
        ]
    }
```

RadiantDeployer/Models/Credentials.swift

```
import Foundation
```

```swift
struct CredentialSet: Identifiable, Codable {
    let id: String
    var name: String
    var accessKeyId: String
    var secretAccessKey: String
    var region: String
    var accountId: String?
    var environment: CredentialEnvironment
    var createdAt: Date
    var lastValidatedAt: Date?
    var isValid: Bool?

    var maskedSecretKey: String {
        guard secretAccessKey.count > 8 else { return "********" }
        let prefix = String(secretAccessKey.prefix(4))
        let suffix = String(secretAccessKey.suffix(4))
        return "\(prefix)...\(suffix)"
    }
}

enum CredentialEnvironment: String, Codable, CaseIterable {
    case dev = "Development"
    case staging = "Staging"
    case prod = "Production"
    case shared = "Shared"
}

struct AWSAccount: Codable {
    let accountId: String
    let accountAlias: String?
    let regions: [String]
}
```

RadiantDeployer/Models/Deployment.swift

```swift
import Foundation

// MARK: - Version Constant

/// Current RADIANT version - matches @radiant/shared/constants/version.ts
let RADIANT_VERSION = "4.17.0"

// MARK: - Deployment Progress

struct DeploymentProgress: Identifiable, Sendable {
    let id = UUID()
```

```
    var phase: DeploymentPhase
    var progress: Double // 0.0 - 1.0
    var currentStack: String?
    var message: String?
    var startedAt: Date
    var estimatedCompletion: Date?
}

enum DeploymentPhase: String, CaseIterable, Sendable {
    case idle = "Idle"
    case validating = "Validating Credentials"
    case bootstrapping = "Bootstrapping CDK"
    case synthesizing = "Synthesizing Stacks"
    case deployingFoundation = "Deploying Foundation"
    case deployingNetworking = "Deploying Networking"
    case deploySecurity = "Deploying Security"
    case deployingData = "Deploying Data Layer"
    case deployingAI = "Deploying AI Services"
    case deployingAPI = "Deploying API Layer"
    case deployingAdmin = "Deploying Admin Dashboard"
    case runningMigrations = "Running Migrations"
    case seedingData = "Seeding Initial Data"
    case verifying = "Verifying Deployment"
    case complete = "Complete"
    case failed = "Failed"

    var progress: Double {
        switch self {
        case .idle: return 0.0
        case .validating: return 0.05
        case .bootstrapping: return 0.10
        case .synthesizing: return 0.15
        case .deployingFoundation: return 0.25
        case .deployingNetworking: return 0.35
        case .deploySecurity: return 0.45
        case .deployingData: return 0.55
        case .deployingAI: return 0.65
        case .deployingAPI: return 0.75
        case .deployingAdmin: return 0.85
        case .runningMigrations: return 0.90
        case .seedingData: return 0.95
        case .verifying: return 0.98
        case .complete: return 1.0
        case .failed: return 0.0
        }
    }
```

```
    var icon: String {
        switch self {
        case .idle: return "circle"
        case .validating: return "checkmark.shield"
        case .bootstrapping: return "arrow.up.circle"
        case .synthesizing: return "doc.text"
        case .deployingFoundation: return "building"
        case .deployingNetworking: return "network"
        case .deploySecurity: return "lock.shield"
        case .deployingData: return "cylinder"
        case .deployingAI: return "cpu"
        case .deployingAPI: return "server.rack"
        case .deployingAdmin: return "rectangle.3.group"
        case .runningMigrations: return "arrow.triangle.2.circlepath"
        case .seedingData: return "leaf"
        case .verifying: return "checkmark.circle"
        case .complete: return "checkmark.circle.fill"
        case .failed: return "xmark.circle.fill"
        }
    }
}

struct DeploymentResult: Identifiable, Codable, Sendable {
    let id: String
    let appId: String
    let environment: String
    let version: String
    let success: Bool
    let startedAt: Date
    let completedAt: Date
    let outputs: DeploymentOutputs?
    let errors: [String]?

    /// Create result with current RADIANT version
    static func create(
        appId: String,
        environment: String,
        success: Bool,
        startedAt: Date,
        outputs: DeploymentOutputs? = nil,
        errors: [String]? = nil
    ) -> DeploymentResult {
        DeploymentResult(
            id: UUID().uuidString,
            appId: appId,
```

```
            environment: environment,
            version: RADIANT_VERSION,
            success: success,
            startedAt: startedAt,
            completedAt: Date(),
            outputs: outputs,
            errors: errors
        )
    }
}

struct DeploymentOutputs: Codable, Sendable {
    let apiUrl: String
    let graphqlUrl: String
    let dashboardUrl: String
    let cognitoUserPoolId: String
    let cognitoClientId: String
    let cognitoDomain: String
    let auroraEndpoint: String
    let s3MediaBucket: String
    let cloudfrontDistribution: String
}

struct LogEntry: Identifiable, Sendable {
    let id = UUID()
    let timestamp: Date
    let level: LogLevel
    let message: String
    let metadata: [String: String]?
}

enum LogLevel: String, Sendable {
    case debug, info, warn, error, success

    var color: String {
        switch self {
        case .debug: return "gray"
        case .info: return "blue"
        case .warn: return "orange"
        case .error: return "red"
        case .success: return "green"
        }
    }

    var icon: String {
        switch self {
```

```
        case .debug: return "ant"
        case .info: return "info.circle"
        case .warn: return "exclamationmark.triangle"
        case .error: return "xmark.circle"
        case .success: return "checkmark.circle"
        }
    }
}
```

RadiantDeployer/Services/CredentialService.swift

```
import Foundation
import SQLite3

/// Manages encrypted credential storage using SQLCipher
actor CredentialService {
    private var db: OpaquePointer?
    private let dbPath: URL
    private let encryptionKey: String

    init() {
        let appSupport = FileManager.default.urls(for: .applicationSupportDirectory, in: .us
        let radiantDir = appSupport.appendingPathComponent("RadiantDeployer", isDirectory: t

        try? FileManager.default.createDirectory(at: radiantDir, withIntermediateDirectories

        dbPath = radiantDir.appendingPathComponent("credentials.db")

        // In production, derive from macOS Keychain
        encryptionKey = Self.getOrCreateEncryptionKey()
    }

    private static func getOrCreateEncryptionKey() -> String {
        let service = "YOUR_ORG_IDENTIFIER.radiant-deployer"
        let account = "db-encryption-key"

        // Try to get existing key
        let query: [String: Any] = [
            kSecClass as String: kSecClassGenericPassword,
            kSecAttrService as String: service,
            kSecAttrAccount as String: account,
            kSecReturnData as String: true
        ]

        var result: AnyObject?
        let status = SecItemCopyMatching(query as CFDictionary, &result)
```

```
        if status == errSecSuccess, let data = result as? Data, let key = String(data: data,
            return key
        }

        // Generate new key
        let newKey = UUID().uuidString + UUID().uuidString
        let keyData = newKey.data(using: .utf8)!

        let addQuery: [String: Any] = [
            kSecClass as String: kSecClassGenericPassword,
            kSecAttrService as String: service,
            kSecAttrAccount as String: account,
            kSecValueData as String: keyData,
            kSecAttrAccessible as String: kSecAttrAccessibleWhenUnlockedThisDeviceOnly
        ]

        SecItemAdd(addQuery as CFDictionary, nil)

        return newKey
    }

    private func openDatabase() throws {
        guard db == nil else { return }

        let flags = SQLITE_OPEN_CREATE | SQLITE_OPEN_READWRITE | SQLITE_OPEN_FULLMUTEX

        guard sqlite3_open_v2(dbPath.path, &db, flags, nil) == SQLITE_OK else {
            throw CredentialError.databaseOpenFailed
        }

        // Set encryption key (SQLCipher)
        let keySQL = "PRAGMA key = '\(encryptionKey)';"
        guard sqlite3_exec(db, keySQL, nil, nil, nil) == SQLITE_OK else {
            throw CredentialError.encryptionFailed
        }

        // Create tables
        let createSQL = """
            CREATE TABLE IF NOT EXISTS credentials (
                id TEXT PRIMARY KEY,
                name TEXT NOT NULL,
                access_key_id TEXT NOT NULL,
                secret_access_key TEXT NOT NULL,
                region TEXT NOT NULL,
                account_id TEXT,
```

```
            environment TEXT NOT NULL,
            created_at TEXT NOT NULL,
            last_validated_at TEXT,
            is_valid INTEGER
        );
    """

    guard sqlite3_exec(db, createSQL, nil, nil, nil) == SQLITE_OK else {
        throw CredentialError.tableCreationFailed
    }
}

func loadCredentials() async throws -> [CredentialSet] {
    try openDatabase()

    let query = "SELECT * FROM credentials ORDER BY created_at DESC;"
    var statement: OpaquePointer?

    guard sqlite3_prepare_v2(db, query, -1, &statement, nil) == SQLITE_OK else {
        throw CredentialError.queryFailed
    }

    defer { sqlite3_finalize(statement) }

    var credentials: [CredentialSet] = []
    let dateFormatter = ISO8601DateFormatter()

    while sqlite3_step(statement) == SQLITE_ROW {
        let id = String(cString: sqlite3_column_text(statement, 0))
        let name = String(cString: sqlite3_column_text(statement, 1))
        let accessKeyId = String(cString: sqlite3_column_text(statement, 2))
        let secretAccessKey = String(cString: sqlite3_column_text(statement, 3))
        let region = String(cString: sqlite3_column_text(statement, 4))
        let accountId = sqlite3_column_text(statement, 5).map { String(cString: $0) }
        let environment = String(cString: sqlite3_column_text(statement, 6))
        let createdAtStr = String(cString: sqlite3_column_text(statement, 7))
        let lastValidatedAtStr = sqlite3_column_text(statement, 8).map { String(cString
        let isValid = sqlite3_column_type(statement, 9) != SQLITE_NULL ? sqlite3_column_

        credentials.append(CredentialSet(
            id: id,
            name: name,
            accessKeyId: accessKeyId,
            secretAccessKey: secretAccessKey,
            region: region,
            accountId: accountId,
```

```
                environment: CredentialEnvironment(rawValue: environment) ?? .shared,
                createdAt: dateFormatter.date(from: createdAtStr) ?? Date(),
                lastValidatedAt: lastValidatedAtStr.flatMap { dateFormatter.date(from: $0) },
                isValid: isValid
            ))
        }

        return credentials
    }

    func saveCredential(_ credential: CredentialSet) async throws {
        try openDatabase()

        let dateFormatter = ISO8601DateFormatter()

        let upsertSQL = """
            INSERT OR REPLACE INTO credentials
            (id, name, access_key_id, secret_access_key, region, account_id, environment, cr
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
        """

        var statement: OpaquePointer?
        guard sqlite3_prepare_v2(db, upsertSQL, -1, &statement, nil) == SQLITE_OK else {
            throw CredentialError.saveFailed
        }

        defer { sqlite3_finalize(statement) }

        sqlite3_bind_text(statement, 1, credential.id, -1, nil)
        sqlite3_bind_text(statement, 2, credential.name, -1, nil)
        sqlite3_bind_text(statement, 3, credential.accessKeyId, -1, nil)
        sqlite3_bind_text(statement, 4, credential.secretAccessKey, -1, nil)
        sqlite3_bind_text(statement, 5, credential.region, -1, nil)

        if let accountId = credential.accountId {
            sqlite3_bind_text(statement, 6, accountId, -1, nil)
        } else {
            sqlite3_bind_null(statement, 6)
        }

        sqlite3_bind_text(statement, 7, credential.environment.rawValue, -1, nil)
        sqlite3_bind_text(statement, 8, dateFormatter.string(from: credential.createdAt), -1

        if let lastValidated = credential.lastValidatedAt {
            sqlite3_bind_text(statement, 9, dateFormatter.string(from: lastValidated), -1, n
        } else {
```

```
            sqlite3_bind_null(statement, 9)
        }

        if let isValid = credential.isValid {
            sqlite3_bind_int(statement, 10, isValid ? 1 : 0)
        } else {
            sqlite3_bind_null(statement, 10)
        }

        guard sqlite3_step(statement) == SQLITE_DONE else {
            throw CredentialError.saveFailed
        }
    }

    func deleteCredential(id: String) async throws {
        try openDatabase()

        let deleteSQL = "DELETE FROM credentials WHERE id = ?;"
        var statement: OpaquePointer?

        guard sqlite3_prepare_v2(db, deleteSQL, -1, &statement, nil) == SQLITE_OK else {
            throw CredentialError.deleteFailed
        }

        defer { sqlite3_finalize(statement) }

        sqlite3_bind_text(statement, 1, id, -1, nil)

        guard sqlite3_step(statement) == SQLITE_DONE else {
            throw CredentialError.deleteFailed
        }
    }

    deinit {
        if db != nil {
            sqlite3_close(db)
        }
    }
}

enum CredentialError: Error, LocalizedError {
    case databaseOpenFailed
    case encryptionFailed
    case tableCreationFailed
    case queryFailed
    case saveFailed
```

```
    case deleteFailed
    case validationFailed(String)

    var errorDescription: String? {
        switch self {
        case .databaseOpenFailed: return "Failed to open credential database"
        case .encryptionFailed: return "Failed to set database encryption"
        case .tableCreationFailed: return "Failed to create database tables"
        case .queryFailed: return "Failed to query credentials"
        case .saveFailed: return "Failed to save credential"
        case .deleteFailed: return "Failed to delete credential"
        case .validationFailed(let reason): return "Credential validation failed: \(reason)"
        }
    }
}
```

RadiantDeployer/Services/CDKService.swift

```
import Foundation
import Combine

/// Manages CDK deployments from bundled infrastructure code
actor CDKService {
    private var currentProcess: Process?
    private let bundledNodePath: URL
    private let bundledInfraPath: URL

    init() {
        let bundle = Bundle.main
        bundledNodePath = bundle.resourceURL!.appendingPathComponent("NodeRuntime/bin/node")
        bundledInfraPath = bundle.resourceURL!.appendingPathComponent("Infrastructure")
    }

    func deploy(
        app: ManagedApp,
        environment: Environment,
        credentials: CredentialSet,
        onPhase: @escaping (DeploymentPhase) -> Void,
        onLog: @escaping (LogEntry) -> Void,
        onProgress: @escaping (Double) -> Void
    ) async throws -> DeploymentResult {
        let startTime = Date()
        var outputs: DeploymentOutputs?
        var errors: [String] = []

        // Create temporary working directory
```

```
let workDir = FileManager.default.temporaryDirectory.appendingPathComponent(UUID().u
try FileManager.default.createDirectory(at: workDir, withIntermediateDirectories: tr

defer {
    try? FileManager.default.removeItem(at: workDir)
}

// Copy infrastructure to working directory
let infraWorkDir = workDir.appendingPathComponent("infrastructure")
try FileManager.default.copyItem(at: bundledInfraPath, to: infraWorkDir)

do {
    // Phase 1: Validate credentials
    onPhase(.validating)
    onLog(LogEntry(timestamp: Date(), level: .info, message: "Validating AWS credent
    try await validateCredentials(credentials)
    onProgress(0.05)
    onLog(LogEntry(timestamp: Date(), level: .success, message: "Credentials validat

    // Phase 2: Install dependencies
    onPhase(.bootstrapping)
    onLog(LogEntry(timestamp: Date(), level: .info, message: "Installing CDK depende
    try await runCommand("npm", args: ["ci"], in: infraWorkDir, credentials: credent
    onProgress(0.10)

    // Phase 3: Bootstrap CDK
    onLog(LogEntry(timestamp: Date(), level: .info, message: "Bootstrapping CDK...",
    try await runCDK(
        command: "bootstrap",
        args: ["aws://\(credentials.accountId ?? "")/\(credentials.region)"],
        in: infraWorkDir,
        credentials: credentials,
        context: buildContext(app: app, environment: environment),
        onLog: onLog
    )
    onProgress(0.15)

    // Phase 4: Synthesize
    onPhase(.synthesizing)
    onLog(LogEntry(timestamp: Date(), level: .info, message: "Synthesizing CloudForm
    try await runCDK(
        command: "synth",
        args: ["--all"],
        in: infraWorkDir,
        credentials: credentials,
        context: buildContext(app: app, environment: environment),
```

```
    onLog: onLog
)
onProgress(0.20)

// Phase 5-11: Deploy stacks
let stacks = [
    ("Foundation", DeploymentPhase.deployingFoundation, 0.30),
    ("Networking", DeploymentPhase.deployingNetworking, 0.40),
    ("Security", DeploymentPhase.deploySecurity, 0.50),
    ("Data", DeploymentPhase.deployingData, 0.60),
    ("AI", DeploymentPhase.deployingAI, 0.70),
    ("API", DeploymentPhase.deployingAPI, 0.80),
    ("Admin", DeploymentPhase.deployingAdmin, 0.90)
]

for (stackName, phase, progress) in stacks {
    onPhase(phase)
    onLog(LogEntry(timestamp: Date(), level: .info, message: "Deploying \(stackN

    let stackOutputs = try await runCDK(
        command: "deploy",
        args: ["\(app.id)-\(environment.rawValue)-\(stackName.lowercased())", "-
        in: infraWorkDir,
        credentials: credentials,
        context: buildContext(app: app, environment: environment),
        onLog: onLog
    )

    onProgress(progress)
    onLog(LogEntry(timestamp: Date(), level: .success, message: "\(stackName) st
}

// Phase 12: Run migrations
onPhase(.runningMigrations)
onLog(LogEntry(timestamp: Date(), level: .info, message: "Running database migra
try await runMigrations(app: app, environment: environment, credentials: credent
onProgress(0.95)

// Phase 13: Verify
onPhase(.verifying)
onLog(LogEntry(timestamp: Date(), level: .info, message: "Verifying deployment..
outputs = try await verifyDeployment(app: app, environment: environment, workDir
onProgress(1.0)

onPhase(.complete)
onLog(LogEntry(timestamp: Date(), level: .success, message: "Deployment complete
```

```
        } catch {
            errors.append(error.localizedDescription)
            onPhase(.failed)
            onLog(LogEntry(timestamp: Date(), level: .error, message: "Deployment failed: \(
        }

        return DeploymentResult.create(
            appId: app.id,
            environment: environment.rawValue,
            success: errors.isEmpty,
            startedAt: startTime,
            outputs: outputs,
            errors: errors.isEmpty ? nil : errors
        )
    }

    private func validateCredentials(_ credentials: CredentialSet) async throws {
        // Find AWS CLI - check multiple locations for compatibility
        let awsPaths = [
            "/opt/homebrew/bin/aws",      // Homebrew on Apple Silicon
            "/usr/local/bin/aws",          // Homebrew on Intel
            "/usr/bin/aws",                // System install
        ]

        guard let awsPath = awsPaths.first(where: { FileManager.default.fileExists(atPath: $
            throw DeploymentError.awsCliNotFound
        }

        let process = Process()
        process.executableURL = URL(fileURLWithPath: awsPath)
        process.arguments = ["sts", "get-caller-identity"]
        process.environment = buildAWSEnvironment(credentials)

        let pipe = Pipe()
        process.standardOutput = pipe
        process.standardError = pipe

        try process.run()
        process.waitUntilExit()

        guard process.terminationStatus == 0 else {
            throw DeploymentError.credentialValidationFailed
        }
    }
```

```
private func runCommand(
    _ command: String,
    args: [String],
    in directory: URL,
    credentials: CredentialSet,
    onLog: @escaping (LogEntry) -> Void
) async throws {
    let process = Process()
    process.executableURL = URL(fileURLWithPath: "/usr/bin/env")
    process.arguments = [command] + args
    process.currentDirectoryURL = directory
    process.environment = buildAWSEnvironment(credentials)

    let pipe = Pipe()
    process.standardOutput = pipe
    process.standardError = pipe

    pipe.fileHandleForReading.readabilityHandler = { handle in
        let data = handle.availableData
        if let output = String(data: data, encoding: .utf8), !output.isEmpty {
            onLog(LogEntry(timestamp: Date(), level: .debug, message: output.trimmingCha
        }
    }

    try process.run()
    process.waitUntilExit()

    pipe.fileHandleForReading.readabilityHandler = nil

    guard process.terminationStatus == 0 else {
        throw DeploymentError.commandFailed(command, Int(process.terminationStatus))
    }
}

@discardableResult
private func runCDK(
    command: String,
    args: [String],
    in directory: URL,
    credentials: CredentialSet,
    context: [String: String],
    onLog: @escaping (LogEntry) -> Void
) async throws -> [String: Any]? {
    var allArgs = [command] + args

    for (key, value) in context {
```

```
            allArgs.append("--context")
            allArgs.append("\(key)=\(value)")
        }

        try await runCommand("npx", args: ["cdk"] + allArgs, in: directory, credentials: cre

        // Read outputs if available
        let outputsFile = directory.appendingPathComponent("outputs.json")
        if FileManager.default.fileExists(atPath: outputsFile.path),
           let data = try? Data(contentsOf: outputsFile),
           let json = try? JSONSerialization.jsonObject(with: data) as? [String: Any] {
            return json
        }

        return nil
    }

    private func buildContext(app: ManagedApp, environment: Environment) -> [String: String]
        let tier = app.environments[environment].tier
        return [
            "appId": app.id,
            "appName": app.name,
            "domain": app.domain,
            "environment": environment.rawValue.lowercased(),
            "tier": String(tier)
        ]
    }

    private func buildAWSEnvironment(_ credentials: CredentialSet) -> [String: String] {
        var env = ProcessInfo.processInfo.environment
        env["AWS_ACCESS_KEY_ID"] = credentials.accessKeyId
        env["AWS_SECRET_ACCESS_KEY"] = credentials.secretAccessKey
        env["AWS_DEFAULT_REGION"] = credentials.region
        env["AWS_REGION"] = credentials.region
        return env
    }

    private func runMigrations(
        app: ManagedApp,
        environment: Environment,
        credentials: CredentialSet,
        onLog: @escaping (LogEntry) -> Void
    ) async throws {
        // Invoke migration Lambda
        onLog(LogEntry(timestamp: Date(), level: .info, message: "Invoking migration Lambda
        // Implementation depends on how migrations are structured
```

```
    }

    private func verifyDeployment(
        app: ManagedApp,
        environment: Environment,
        workDir: URL
    ) async throws -> DeploymentOutputs {
        // Read outputs from CDK deployment
        let outputsFile = workDir.appendingPathComponent("outputs.json")

        guard FileManager.default.fileExists(atPath: outputsFile.path),
              let data = try? Data(contentsOf: outputsFile),
              let json = try? JSONSerialization.jsonObject(with: data) as? [String: Any] els
            throw DeploymentError.outputsNotFound
        }

        // Parse outputs (structure depends on CDK output format)
        // This is a simplified version
        return DeploymentOutputs(
            apiUrl: json["ApiUrl"] as? String ?? "",
            graphqlUrl: json["GraphQLUrl"] as? String ?? "",
            dashboardUrl: json["DashboardUrl"] as? String ?? "",
            cognitoUserPoolId: json["CognitoUserPoolId"] as? String ?? "",
            cognitoClientId: json["CognitoClientId"] as? String ?? "",
            cognitoDomain: json["CognitoDomain"] as? String ?? "",
            auroraEndpoint: json["AuroraEndpoint"] as? String ?? "",
            s3MediaBucket: json["S3MediaBucket"] as? String ?? "",
            cloudfrontDistribution: json["CloudFrontDistribution"] as? String ?? ""
        )
    }

    func cancel() {
        currentProcess?.terminate()
    }
}

enum DeploymentError: Error, LocalizedError {
    case credentialValidationFailed
    case commandFailed(String, Int)
    case outputsNotFound
    case verificationFailed(String)
    case awsCliNotFound
    case nodeNotFound
    case cdkBootstrapFailed

    var errorDescription: String? {
```

```
        switch self {
        case .credentialValidationFailed:
            return "AWS credential validation failed"
        case .commandFailed(let cmd, let code):
            return "Command '\(cmd)' failed with exit code \(code)"
        case .outputsNotFound:
            return "Deployment outputs not found"
        case .verificationFailed(let reason):
            return "Deployment verification failed: \(reason)"
        case .awsCliNotFound:
            return "AWS CLI not found. Install via: brew install awscli"
        case .nodeNotFound:
            return "Node.js not found in bundled resources"
        case .cdkBootstrapFailed:
            return "CDK bootstrap failed. Check AWS permissions."
        }
    }
}
```

RadiantDeployer/Services/AWSService.swift

```
import Foundation

/// Direct AWS API interactions (for validation, health checks, etc.)
actor AWSService {

    func validateCredentials(_ credentials: CredentialSet) async throws -> AWSAccount {
        let process = Process()
        process.executableURL = URL(fileURLWithPath: "/usr/local/bin/aws")
        process.arguments = ["sts", "get-caller-identity", "--output", "json"]

        var env = ProcessInfo.processInfo.environment
        env["AWS_ACCESS_KEY_ID"] = credentials.accessKeyId
        env["AWS_SECRET_ACCESS_KEY"] = credentials.secretAccessKey
        env["AWS_DEFAULT_REGION"] = credentials.region
        process.environment = env

        let pipe = Pipe()
        process.standardOutput = pipe
        process.standardError = Pipe()

        try process.run()
        process.waitUntilExit()

        guard process.terminationStatus == 0 else {
            throw AWSError.credentialValidationFailed
```

```
        }

        let data = pipe.fileHandleForReading.readDataToEndOfFile()
        guard let json = try? JSONSerialization.jsonObject(with: data) as? [String: Any],
              let accountId = json["Account"] as? String else {
            throw AWSError.invalidResponse
        }

        // Get account alias
        let alias = try? await getAccountAlias(credentials: credentials)

        return AWSAccount(
            accountId: accountId,
            accountAlias: alias,
            regions: ["us-east-1", "us-west-2", "eu-west-1"] // Could be dynamic
        )
    }

    private func getAccountAlias(credentials: CredentialSet) async throws -> String? {
        let process = Process()
        process.executableURL = URL(fileURLWithPath: "/usr/local/bin/aws")
        process.arguments = ["iam", "list-account-aliases", "--output", "json"]

        var env = ProcessInfo.processInfo.environment
        env["AWS_ACCESS_KEY_ID"] = credentials.accessKeyId
        env["AWS_SECRET_ACCESS_KEY"] = credentials.secretAccessKey
        env["AWS_DEFAULT_REGION"] = credentials.region
        process.environment = env

        let pipe = Pipe()
        process.standardOutput = pipe
        process.standardError = Pipe()

        try process.run()
        process.waitUntilExit()

        guard process.terminationStatus == 0 else { return nil }

        let data = pipe.fileHandleForReading.readDataToEndOfFile()
        guard let json = try? JSONSerialization.jsonObject(with: data) as? [String: Any],
              let aliases = json["AccountAliases"] as? [String],
              let alias = aliases.first else {
            return nil
        }

        return alias
```

```
    }

    func checkHealth(app: ManagedApp, environment: Environment, credentials: CredentialSet)
        // Check API Gateway health endpoint
        guard let apiUrl = app.environments[environment].apiUrl else {
            return .unknown
        }

        guard let url = URL(string: "\(apiUrl)/health") else {
            return .unknown
        }

        var request = URLRequest(url: url)
        request.timeoutInterval = 10

        do {
            let (_, response) = try await URLSession.shared.data(for: request)
            guard let httpResponse = response as? HTTPURLResponse else {
                return .unknown
            }

            switch httpResponse.statusCode {
            case 200: return .healthy
            case 503: return .degraded
            default: return .unhealthy
            }
        } catch {
            return .unhealthy
        }
    }
}

enum AWSError: Error, LocalizedError {
    case credentialValidationFailed
    case invalidResponse
    case apiError(String)

    var errorDescription: String? {
        switch self {
        case .credentialValidationFailed:
            return "AWS credential validation failed"
        case .invalidResponse:
            return "Invalid response from AWS"
        case .apiError(let message):
            return "AWS API error: \(message)"
        }
```

```
    }
}
```

RadiantDeployer/Services/APIService.swift

```swift
import Foundation

/// Communicates with deployed RADIANT APIs
actor APIService {
    private let session = URLSession.shared
    private var authToken: String?

    func setAuthToken(_ token: String) {
        self.authToken = token
    }

    func fetchProviders(baseUrl: String) async throws -> [Provider] {
        let url = URL(string: "\(baseUrl)/api/v2/providers")!
        return try await fetch(url: url)
    }

    func fetchModels(baseUrl: String) async throws -> [Model] {
        let url = URL(string: "\(baseUrl)/api/v2/models")!
        return try await fetch(url: url)
    }

    private func fetch<T: Decodable>(url: URL) async throws -> T {
        var request = URLRequest(url: url)
        request.setValue("application/json", forHTTPHeaderField: "Content-Type")

        if let token = authToken {
            request.setValue("Bearer \(token)", forHTTPHeaderField: "Authorization")
        }

        let (data, response) = try await session.data(for: request)

        guard let httpResponse = response as? HTTPURLResponse,
              (200...299).contains(httpResponse.statusCode) else {
            throw APIError.requestFailed
        }

        return try JSONDecoder().decode(T.self, from: data)
    }
}

// Placeholder types for API responses
```

```swift
struct Provider: Codable, Identifiable {
    let id: String
    let name: String
    let status: String
}

struct Model: Codable, Identifiable {
    let id: String
    let name: String
    let providerId: String
}

enum APIError: Error {
    case requestFailed
    case decodingFailed
}
```

PART 4: SWIFT VIEWS

RadiantDeployer/Views/MainView.swift

```swift
import SwiftUI

struct MainView: View {
    @EnvironmentObject var appState: AppState

    var body: some View {
        NavigationSplitView {
            Sidebar()
        } detail: {
            DetailView()
        }
        .navigationSplitViewStyle(.balanced)
        .alert(item: $appState.error) { error in
            Alert(
                title: Text("Error"),
                message: Text(error.localizedDescription),
                dismissButton: .default(Text("OK"))
            )
        }
    }
}

struct Sidebar: View {
    @EnvironmentObject var appState: AppState
```

```
    var body: some View {
        List(selection: $appState.selectedTab) {
            Section("Navigation") {
                ForEach(NavigationTab.allCases) { tab in
                    Label(tab.rawValue, systemImage: tab.icon)
                        .tag(tab)
                }
            }

            if !appState.apps.isEmpty {
                Section("Apps") {
                    ForEach(appState.apps) { app in
                        AppRow(app: app)
                    }
                }
            }
        }
        .listStyle(.sidebar)
        .frame(minWidth: 220)
        .toolbar {
            ToolbarItem(placement: .automatic) {
                Button(action: { /* Add app */ }) {
                    Image(systemName: "plus")
                }
            }
        }
    }
}

struct AppRow: View {
    @EnvironmentObject var appState: AppState
    let app: ManagedApp

    var body: some View {
        HStack {
            VStack(alignment: .leading, spacing: 2) {
                Text(app.name)
                    .font(.headline)
                Text(app.domain)
                    .font(.caption)
                    .foregroundColor(.secondary)
            }

            Spacer()
```

```
            HStack(spacing: 4) {
                EnvironmentDot(status: app.environments.dev)
                EnvironmentDot(status: app.environments.staging)
                EnvironmentDot(status: app.environments.prod)
            }
        }
        .padding(.vertical, 4)
        .contentShape(Rectangle())
        .onTapGesture {
            appState.selectedApp = app
            appState.selectedTab = .deploy
        }
    }
}

struct EnvironmentDot: View {
    let status: EnvironmentStatus

    var body: some View {
        Circle()
            .fill(status.deployed ? Color(status.healthStatus.color) : Color.gray.opacity(0.
            .frame(width: 8, height: 8)
    }
}

struct DetailView: View {
    @EnvironmentObject var appState: AppState

    var body: some View {
        Group {
            switch appState.selectedTab {
            case .apps:
                AppsView()
            case .deploy:
                DeployView()
            case .providers:
                ProvidersView()
            case .models:
                ModelsView()
            case .settings:
                SettingsView()
            }
        }
    }
}
```

RadiantDeployer/Views/AppsView.swift

```swift
import SwiftUI

struct AppsView: View {
    @EnvironmentObject var appState: AppState
    @State private var showingAddApp = false

    var body: some View {
        ScrollView {
            LazyVGrid(columns: [GridItem(.adaptive(minimum: 300))], spacing: 20) {
                ForEach(appState.apps) { app in
                    AppCard(app: app)
                }

                AddAppCard(action: { showingAddApp = true })
            }
            .padding()
        }
        .navigationTitle("Applications")
        .sheet(isPresented: $showingAddApp) {
            AddAppSheet()
        }
    }
}

struct AppCard: View {
    @EnvironmentObject var appState: AppState
    let app: ManagedApp

    var body: some View {
        VStack(alignment: .leading, spacing: 16) {
            HStack {
                VStack(alignment: .leading) {
                    Text(app.name)
                        .font(.title2)
                        .fontWeight(.semibold)
                    Text(app.domain)
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                }

                Spacer()

                Menu {
                    Button("Edit") { }
```

```
                    Button("View Logs") { }
                    Divider()
                    Button("Delete", role: .destructive) { }
                } label: {
                    Image(systemName: "ellipsis.circle")
                        .font(.title3)
                }
            }

            Divider()

            HStack(spacing: 20) {
                EnvironmentStatusView(name: "DEV", status: app.environments.dev, color: .blu
                EnvironmentStatusView(name: "STAGING", status: app.environments.staging, col
                EnvironmentStatusView(name: "PROD", status: app.environments.prod, color: .g
            }

            HStack {
                Button("Deploy") {
                    appState.selectedApp = app
                    appState.selectedTab = .deploy
                }
                .buttonStyle(.borderedProminent)

                Button("Dashboard") {
                    if let url = URL(string: app.environments.prod.dashboardUrl ?? "") {
                        NSWorkspace.shared.open(url)
                    }
                }
                .buttonStyle(.bordered)
                .disabled(app.environments.prod.dashboardUrl == nil)
            }
        }
        .padding()
        .background(Color(.windowBackgroundColor))
        .cornerRadius(12)
        .shadow(radius: 2)
    }
}

struct EnvironmentStatusView: View {
    let name: String
    let status: EnvironmentStatus
    let color: Color

    var body: some View {
```

```
            VStack(spacing: 4) {
                Text(name)
                    .font(.caption)
                    .fontWeight(.medium)
                    .foregroundColor(.secondary)

                Circle()
                    .fill(status.deployed ? Color(status.healthStatus.color) : Color.gray.opacit
                    .frame(width: 12, height: 12)

                if status.deployed, let version = status.version {
                    Text("v\(version)")
                        .font(.caption2)
                        .foregroundColor(.secondary)
                }
            }
        }
    }

    struct AddAppCard: View {
        let action: () -> Void

        var body: some View {
            Button(action: action) {
                VStack(spacing: 12) {
                    Image(systemName: "plus.circle")
                        .font(.system(size: 40))
                        .foregroundColor(.accentColor)
                    Text("Add Application")
                        .font(.headline)
                }
                .frame(maxWidth: .infinity, minHeight: 150)
                .background(Color(.windowBackgroundColor).opacity(0.5))
                .cornerRadius(12)
                .overlay(
                    RoundedRectangle(cornerRadius: 12)
                        .stroke(style: StrokeStyle(lineWidth: 2, dash: [8]))
                        .foregroundColor(.secondary.opacity(0.3))
                )
            }
            .buttonStyle(.plain)
        }
    }

    struct AddAppSheet: View {
        @Environment(\.dismiss) var dismiss
```

```
    @State private var appId = ""
    @State private var appName = ""
    @State private var domain = ""

    var body: some View {
        VStack(spacing: 20) {
            Text("Add New Application")
                .font(.title2)
                .fontWeight(.semibold)

            Form {
                TextField("App ID (lowercase, no spaces)", text: $appId)
                TextField("App Name", text: $appName)
                TextField("Domain (e.g., myapp.YOUR_DOMAIN.com)", text: $domain)
            }
            .formStyle(.grouped)

            HStack {
                Button("Cancel") { dismiss() }
                    .buttonStyle(.bordered)

                Button("Add Application") {
                    // Add app logic
                    dismiss()
                }
                .buttonStyle(.borderedProminent)
                .disabled(appId.isEmpty || appName.isEmpty || domain.isEmpty)
            }
        }
        .padding()
        .frame(width: 400)
    }
}
```

RadiantDeployer/Views/DeployView.swift

```
import SwiftUI

struct DeployView: View {
    @EnvironmentObject var appState: AppState
    @State private var selectedCredential: CredentialSet?

    var body: some View {
        HSplitView {
            // Left: Configuration
            DeployConfigPanel(selectedCredential: $selectedCredential)
```

```
                .frame(minWidth: 350, maxWidth: 450)

            // Right: Logs
            DeployLogPanel()
                .frame(minWidth: 500)
        }
        .navigationTitle("Deploy")
    }
}

struct DeployConfigPanel: View {
    @EnvironmentObject var appState: AppState
    @Binding var selectedCredential: CredentialSet?
    @State private var selectedTier: Int = 1

    var body: some View {
        ScrollView {
            VStack(alignment: .leading, spacing: 24) {
                // App Selection
                Section {
                    Picker("Application", selection: $appState.selectedApp) {
                        Text("Select an app").tag(nil as ManagedApp?)
                        ForEach(appState.apps) { app in
                            Text(app.name).tag(app as ManagedApp?)
                        }
                    }
                    .pickerStyle(.menu)
                } header: {
                    Text("Application")
                        .font(.headline)
                }

                // Environment Selection
                Section {
                    Picker("Environment", selection: $appState.selectedEnvironment) {
                        ForEach(Environment.allCases) { env in
                            HStack {
                                Circle()
                                    .fill(env.color)
                                    .frame(width: 8, height: 8)
                                Text(env.rawValue)
                            }
                            .tag(env)
                        }
                    }
                    .pickerStyle(.segmented)
```

```
} header: {
    Text("Environment")
        .font(.headline)
}

// Credentials
Section {
    Picker("AWS Credentials", selection: $selectedCredential) {
        Text("Select credentials").tag(nil as CredentialSet?)
        ForEach(appState.credentials) { cred in
            Text("\(cred.name) (\(cred.environment.rawValue))")
                .tag(cred as CredentialSet?)
        }
    }
    .pickerStyle(.menu)

    if let cred = selectedCredential {
        HStack {
            Label(cred.region, systemImage: "globe")
            Spacer()
            if let valid = cred.isValid {
                Image(systemName: valid ? "checkmark.circle.fill" : "xmark.c
                    .foregroundColor(valid ? .green : .red)
            }
        }
        .font(.caption)
        .foregroundColor(.secondary)
    }

    Button("Manage Credentials") {
        appState.selectedTab = .settings
    }
    .font(.caption)
} header: {
    Text("AWS Credentials")
        .font(.headline)
}

// Tier Selection
Section {
    Picker("Infrastructure Tier", selection: $selectedTier) {
        Text("1 - SEED (Development)").tag(1)
        Text("2 - STARTUP (Small Production)").tag(2)
        Text("3 - GROWTH (Medium Production)").tag(3)
        Text("4 - SCALE (Large + Multi-Region)").tag(4)
        Text("5 - ENTERPRISE (Full Compliance)").tag(5)
```

```
                }
                .pickerStyle(.menu)
            } header: {
                Text("Infrastructure Tier")
                    .font(.headline)
            }

            Divider()

            // Deploy Button
            Button(action: startDeployment) {
                HStack {
                    if appState.isDeploying {
                        ProgressView()
                            .scaleEffect(0.8)
                    } else {
                        Image(systemName: "arrow.up.circle.fill")
                    }
                    Text(appState.isDeploying ? "Deploying..." : "Deploy")
                }
                .frame(maxWidth: .infinity)
            }
            .buttonStyle(.borderedProminent)
            .controlSize(.large)
            .disabled(!canDeploy)

            // Progress
            if let progress = appState.deploymentProgress {
                VStack(alignment: .leading, spacing: 8) {
                    HStack {
                        Image(systemName: progress.phase.icon)
                        Text(progress.phase.rawValue)
                        Spacer()
                        Text("\(Int(progress.progress * 100))%")
                    }
                    .font(.subheadline)

                    ProgressView(value: progress.progress)
                }
                .padding()
                .background(Color(.textBackgroundColor))
                .cornerRadius(8)
            }
        }
        .padding()
    }
```

```
}

private var canDeploy: Bool {
    appState.selectedApp != nil &&
    selectedCredential != nil &&
    !appState.isDeploying
}

private func startDeployment() {
    guard let app = appState.selectedApp,
          let credentials = selectedCredential else { return }

    appState.isDeploying = true
    appState.deploymentLogs = []

    Task {
        let result = try await appState.cdkService.deploy(
            app: app,
            environment: appState.selectedEnvironment,
            credentials: credentials,
            onPhase: { phase in
                Task { @MainActor in
                    appState.deploymentProgress = DeploymentProgress(
                        phase: phase,
                        progress: phase.progress,
                        currentStack: nil,
                        message: nil,
                        startedAt: Date()
                    )
                }
            },
            onLog: { log in
                Task { @MainActor in
                    appState.deploymentLogs.append(log)
                }
            },
            onProgress: { progress in
                Task { @MainActor in
                    appState.deploymentProgress?.progress = progress
                }
            }
        )

        await MainActor.run {
            appState.isDeploying = false
```

```
                if result.success {
                    // Update app status
                    if let index = appState.apps.firstIndex(where: { $0.id == app.id }) {
                        appState.apps[index].environments[appState.selectedEnvironment].depl
                        appState.apps[index].environments[appState.selectedEnvironment].vers
                        appState.apps[index].environments[appState.selectedEnvironment].heal
                        appState.apps[index].environments[appState.selectedEnvironment].apiU
                        appState.apps[index].environments[appState.selectedEnvironment].dash
                    }
                }
            }
        }
    }
}

struct DeployLogPanel: View {
    @EnvironmentObject var appState: AppState

    var body: some View {
        VStack(alignment: .leading, spacing: 0) {
            HStack {
                Text("Deployment Logs")
                    .font(.headline)
                Spacer()
                Button(action: { appState.deploymentLogs = [] }) {
                    Image(systemName: "trash")
                }
                .buttonStyle(.borderless)
                .disabled(appState.deploymentLogs.isEmpty)
            }
            .padding()

            Divider()

            ScrollViewReader { proxy in
                ScrollView {
                    LazyVStack(alignment: .leading, spacing: 4) {
                        ForEach(appState.deploymentLogs) { log in
                            LogRow(log: log)
                                .id(log.id)
                        }
                    }
                    .padding()
                }
                .onChange(of: appState.deploymentLogs.count) { _ in
                    if let lastLog = appState.deploymentLogs.last {
```

```
                        withAnimation {
                            proxy.scrollTo(lastLog.id, anchor: .bottom)
                        }
                    }
                }
            }
            .background(Color(.textBackgroundColor))
        }
    }
}

struct LogRow: View {
    let log: LogEntry

    private var timeString: String {
        let formatter = DateFormatter()
        formatter.dateFormat = "HH:mm:ss.SSS"
        return formatter.string(from: log.timestamp)
    }

    var body: some View {
        HStack(alignment: .top, spacing: 8) {
            Text(timeString)
                .font(.system(.caption, design: .monospaced))
                .foregroundColor(.secondary)

            Image(systemName: log.level.icon)
                .foregroundColor(Color(log.level.color))
                .font(.caption)

            Text(log.message)
                .font(.system(.caption, design: .monospaced))
                .foregroundColor(Color(log.level.color))
                .textSelection(.enabled)
        }
    }
}
```

RadiantDeployer/Views/SettingsView.swift

```
import SwiftUI

struct SettingsView: View {
    @EnvironmentObject var appState: AppState

    var body: some View {
```

```
        TabView {
            CredentialsSettingsView()
                .tabItem {
                    Label("Credentials", systemImage: "key")
                }

            GeneralSettingsView()
                .tabItem {
                    Label("General", systemImage: "gearshape")
                }

            AboutView()
                .tabItem {
                    Label("About", systemImage: "info.circle")
                }
        }
        .padding()
    }
}

struct CredentialsSettingsView: View {
    @EnvironmentObject var appState: AppState
    @State private var showingAddCredential = false
    @State private var selectedCredential: CredentialSet?
    @State private var isValidating = false

    var body: some View {
        VStack(alignment: .leading, spacing: 20) {
            HStack {
                Text("AWS Credentials")
                    .font(.title2)
                    .fontWeight(.semibold)

                Spacer()

                Button(action: { showingAddCredential = true }) {
                    Label("Add Credential", systemImage: "plus")
                }
                .buttonStyle(.borderedProminent)
            }

            List(selection: $selectedCredential) {
                ForEach(appState.credentials) { credential in
                    CredentialRow(credential: credential, isValidating: isValidating && sele
                        .tag(credential)
                }
```

```
                .onDelete(perform: deleteCredentials)
            }
            .listStyle(.inset)

            if let credential = selectedCredential {
                HStack {
                    Button("Validate") {
                        validateCredential(credential)
                    }
                    .disabled(isValidating)

                    Button("Delete", role: .destructive) {
                        deleteCredential(credential)
                    }
                }
            }
        }
        .sheet(isPresented: $showingAddCredential) {
            AddCredentialSheet()
        }
    }

    private func validateCredential(_ credential: CredentialSet) {
        isValidating = true

        Task {
            do {
                let account = try await appState.awsService.validateCredentials(credential)

                await MainActor.run {
                    if let index = appState.credentials.firstIndex(where: { $0.id == credent
                        appState.credentials[index].accountId = account.accountId
                        appState.credentials[index].isValid = true
                        appState.credentials[index].lastValidatedAt = Date()
                    }
                    isValidating = false
                }

                // Save updated credential
                try await appState.credentialService.saveCredential(appState.credentials.fir
            } catch {
                await MainActor.run {
                    if let index = appState.credentials.firstIndex(where: { $0.id == credent
                        appState.credentials[index].isValid = false
                    }
                    isValidating = false
```

```
                    appState.error = AppError(message: "Credential validation failed", under
                }
            }
        }
    }

    private func deleteCredential(_ credential: CredentialSet) {
        Task {
            try await appState.credentialService.deleteCredential(id: credential.id)
            await MainActor.run {
                appState.credentials.removeAll { $0.id == credential.id }
                selectedCredential = nil
            }
        }
    }

    private func deleteCredentials(at offsets: IndexSet) {
        for index in offsets {
            let credential = appState.credentials[index]
            deleteCredential(credential)
        }
    }
}

struct CredentialRow: View {
    let credential: CredentialSet
    let isValidating: Bool

    var body: some View {
        HStack {
            VStack(alignment: .leading, spacing: 4) {
                Text(credential.name)
                    .font(.headline)
                HStack {
                    Text(credential.accessKeyId)
                    Text("Ã¢â¬Â¢")
                    Text(credential.region)
                }
                .font(.caption)
                .foregroundColor(.secondary)
            }

            Spacer()

            if isValidating {
                ProgressView()
```

```
                    .scaleEffect(0.7)
            } else if let isValid = credential.isValid {
                Image(systemName: isValid ? "checkmark.circle.fill" : "xmark.circle.fill")
                    .foregroundColor(isValid ? .green : .red)
            }

            Text(credential.environment.rawValue)
                .font(.caption)
                .padding(.horizontal, 8)
                .padding(.vertical, 4)
                .background(Color.accentColor.opacity(0.1))
                .cornerRadius(4)
        }
        .padding(.vertical, 4)
    }
}

struct AddCredentialSheet: View {
    @EnvironmentObject var appState: AppState
    @Environment(\.dismiss) var dismiss

    @State private var name = ""
    @State private var accessKeyId = ""
    @State private var secretAccessKey = ""
    @State private var region = "us-east-1"
    @State private var environment: CredentialEnvironment = .dev
    @State private var isSaving = false

    var body: some View {
        VStack(spacing: 20) {
            Text("Add AWS Credential")
                .font(.title2)
                .fontWeight(.semibold)

            Form {
                TextField("Name (e.g., 'Production AWS')", text: $name)
                TextField("Access Key ID", text: $accessKeyId)
                SecureField("Secret Access Key", text: $secretAccessKey)

                Picker("Region", selection: $region) {
                    Text("US East (N. Virginia)").tag("us-east-1")
                    Text("US West (Oregon)").tag("us-west-2")
                    Text("Europe (Ireland)").tag("eu-west-1")
                    Text("Europe (Frankfurt)").tag("eu-central-1")
                    Text("Asia Pacific (Tokyo)").tag("ap-northeast-1")
                    Text("Asia Pacific (Singapore)").tag("ap-southeast-1")
```

```
                }

                Picker("Environment", selection: $environment) {
                    ForEach(CredentialEnvironment.allCases, id: \.self) { env in
                        Text(env.rawValue).tag(env)
                    }
                }
            }
            .formStyle(.grouped)

            HStack {
                Button("Cancel") { dismiss() }
                    .buttonStyle(.bordered)

                Button("Add Credential") {
                    saveCredential()
                }
                .buttonStyle(.borderedProminent)
                .disabled(!isValid || isSaving)
            }
        }
        .padding()
        .frame(width: 450)
    }

    private var isValid: Bool {
        !name.isEmpty && !accessKeyId.isEmpty && !secretAccessKey.isEmpty
    }

    private func saveCredential() {
        isSaving = true

        let credential = CredentialSet(
            id: UUID().uuidString,
            name: name,
            accessKeyId: accessKeyId,
            secretAccessKey: secretAccessKey,
            region: region,
            accountId: nil,
            environment: environment,
            createdAt: Date(),
            lastValidatedAt: nil,
            isValid: nil
        )

        Task {
```

```
                try await appState.credentialService.saveCredential(credential)

                await MainActor.run {
                    appState.credentials.append(credential)
                    isSaving = false
                    dismiss()
                }
            }
        }
    }

    struct GeneralSettingsView: View {
        @AppStorage("autoValidateCredentials") private var autoValidate = true
        @AppStorage("showAdvancedOptions") private var showAdvanced = false
        @AppStorage("defaultTier") private var defaultTier = 1

        var body: some View {
            Form {
                Section {
                    Toggle("Auto-validate credentials on launch", isOn: $autoValidate)
                    Toggle("Show advanced deployment options", isOn: $showAdvanced)
                }

                Section {
                    Picker("Default Infrastructure Tier", selection: $defaultTier) {
                        Text("1 - SEED").tag(1)
                        Text("2 - STARTUP").tag(2)
                        Text("3 - GROWTH").tag(3)
                        Text("4 - SCALE").tag(4)
                        Text("5 - ENTERPRISE").tag(5)
                    }
                }
            }
            .formStyle(.grouped)
        }
    }

    struct AboutView: View {
        var body: some View {
            VStack(spacing: 20) {
                Image(systemName: "cloud.fill")
                    .font(.system(size: 80))
                    .foregroundColor(.accentColor)

                Text("RADIANT Deployer")
                    .font(.title)
```

```
                .fontWeight(.bold)

            Text("Version 2.2.0")
                .foregroundColor(.secondary)

            Text("Production-grade multi-tenant AWS SaaS infrastructure deployment and manag
                .multilineTextAlignment(.center)
                .foregroundColor(.secondary)

            Divider()

            VStack(alignment: .leading, spacing: 8) {
                Text("Features:")
                    .font(.headline)

                FeatureRow(icon: "server.rack", text: "Multi-environment deployment (Dev/Sta
                FeatureRow(icon: "cpu", text: "20+ AI providers, 30+ self-hosted models")
                FeatureRow(icon: "lock.shield", text: "HIPAA & SOC 2 compliance")
                FeatureRow(icon: "person.2", text: "Two-person approval for production")
                FeatureRow(icon: "globe", text: "Multi-region global deployment")
            }

            Spacer()

            Text("Ã¸Ã© 2024 Zynapses Inc.")
                .font(.caption)
                .foregroundColor(.secondary)
        }
        .padding()
    }
}

struct FeatureRow: View {
    let icon: String
    let text: String

    var body: some View {
        HStack {
            Image(systemName: icon)
                .foregroundColor(.accentColor)
                .frame(width: 20)
            Text(text)
        }
    }
}
```

RadiantDeployer/Views/ProvidersView.swift

```
import SwiftUI

struct ProvidersView: View {
    @EnvironmentObject var appState: AppState

    var body: some View {
        VStack {
            Text("AI Providers")
                .font(.title)
            Text("Provider management is available in the Admin Dashboard after deployment.'
                .foregroundColor(.secondary)
        }
        .frame(maxWidth: .infinity, maxHeight: .infinity)
        .navigationTitle("Providers")
    }
}
```

RadiantDeployer/Views/ModelsView.swift

```
import SwiftUI

struct ModelsView: View {
    @EnvironmentObject var appState: AppState

    var body: some View {
        VStack {
            Text("AI Models")
                .font(.title)
            Text("Model management is available in the Admin Dashboard after deployment.")
                .foregroundColor(.secondary)
        }
        .frame(maxWidth: .infinity, maxHeight: .infinity)
        .navigationTitle("Models")
    }
}
```

PART 5: BUNDLE SCRIPTS

tools/scripts/bundle-infrastructure.sh

```
#!/bin/bash
set -e
```

```
echo "=== RADIANT Infrastructure Bundler ==="

# Paths
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
ROOT_DIR="$(cd "$SCRIPT_DIR/../.." && pwd)"
INFRA_SOURCE="$ROOT_DIR/packages/infrastructure"
SWIFT_APP="$ROOT_DIR/apps/swift-deployer"
BUNDLE_TARGET="$SWIFT_APP/Resources/Infrastructure"

echo "Source: $INFRA_SOURCE"
echo "Target: $BUNDLE_TARGET"

# Clean and create target
rm -rf "$BUNDLE_TARGET"
mkdir -p "$BUNDLE_TARGET"

# Copy CDK source
echo "Copying CDK source..."
cp -R "$INFRA_SOURCE/bin" "$BUNDLE_TARGET/"
cp -R "$INFRA_SOURCE/lib" "$BUNDLE_TARGET/"
cp -R "$INFRA_SOURCE/lambda" "$BUNDLE_TARGET/"
cp "$INFRA_SOURCE/package.json" "$BUNDLE_TARGET/"
cp "$INFRA_SOURCE/package-lock.json" "$BUNDLE_TARGET/" 2>/dev/null || true
cp "$INFRA_SOURCE/cdk.json" "$BUNDLE_TARGET/"
cp "$INFRA_SOURCE/tsconfig.json" "$BUNDLE_TARGET/"

# Copy migrations
if [ -d "$ROOT_DIR/migrations" ]; then
    echo "Copying migrations..."
    cp -R "$ROOT_DIR/migrations" "$BUNDLE_TARGET/"
fi

# Install production dependencies
echo "Installing production dependencies..."
cd "$BUNDLE_TARGET"
npm ci --production

echo "=== Bundle complete ==="
echo "Size: $(du -sh "$BUNDLE_TARGET" | cut -f1)"
```

tools/scripts/bundle-node.sh

```
#!/bin/bash
set -e

echo "=== Node.js Runtime Bundler ==="
```

```bash
# Paths
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
SWIFT_APP="$(cd "$SCRIPT_DIR/../../apps/swift-deployer" && pwd)"
NODE_TARGET="$SWIFT_APP/Resources/NodeRuntime"

# Node version
NODE_VERSION="20.10.0"

echo "Target: $NODE_TARGET"
echo "Node Version: $NODE_VERSION"

# Clean and create target
rm -rf "$NODE_TARGET"
mkdir -p "$NODE_TARGET"

# Download Node.js for macOS ARM64
ARCH=$(uname -m)
if [ "$ARCH" = "arm64" ]; then
    NODE_ARCH="arm64"
else
    NODE_ARCH="x64"
fi

NODE_URL="https://nodejs.org/dist/v${NODE_VERSION}/node-v${NODE_VERSION}-darwin-${NODE_ARCH}
TEMP_DIR=$(mktemp -d)

echo "Downloading Node.js..."
curl -sL "$NODE_URL" | tar xz -C "$TEMP_DIR"

# Copy only necessary files
echo "Copying runtime..."
cp -R "$TEMP_DIR/node-v${NODE_VERSION}-darwin-${NODE_ARCH}/bin" "$NODE_TARGET/"
cp -R "$TEMP_DIR/node-v${NODE_VERSION}-darwin-${NODE_ARCH}/lib" "$NODE_TARGET/"

# Cleanup
rm -rf "$TEMP_DIR"

echo "=== Bundle complete ==="
echo "Size: $(du -sh "$NODE_TARGET" | cut -f1)"
```

BUILD INSTRUCTIONS

1. **Build Shared Package**:

```
cd packages/shared
npm install
npm run build
```

2. **Bundle Node.js Runtime**:

```
chmod +x tools/scripts/bundle-node.sh
./tools/scripts/bundle-node.sh
```

3. **Bundle Infrastructure** (after completing Prompts 2-3):

```
chmod +x tools/scripts/bundle-infrastructure.sh
./tools/scripts/bundle-infrastructure.sh
```

4. **Open in Xcode**:

```
cd apps/swift-deployer
open RadiantDeployer.xcodeproj
```

5. **Build and Run**:
   - Select "My Mac" as the destination
   - Press Ã¢Å'ËœR to build and run

DEPENDENCIES

Swift (Add to Xcode Project)

- SQLite3 (System framework)
- Security (System framework)

Node.js (bundled)

- aws-cdk: ^2.120.0
- aws-cdk-lib: ^2.120.0
- constructs: ^10.3.0

NEXT PROMPTS

Continue with: - **Prompt 2**: CDK Infrastructure Stacks (VPC, Database, Security, Storage) - **Prompt 3**: CDK AI & API Stacks (LiteLLM, SageMaker, API Gateway) - **Prompt 4**: Lambda Functions - Core - **Prompt 5**: Lambda Functions - Admin & Billing - **Prompt 6**: Self-Hosted Models & Mid-Level Services - **Prompt 7**: External Providers & Database Schema - **Prompt 8**: Admin Web Dashboard - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 1: Foundation & Swift Deployment App RADIANT v2.2.0 - December 2024

â• â•

END OF SECTION 1

â• â•

â• â•

SECTION-02-CDK-INFRASTRUCTURE-STACKS

SECTION 2: CDK INFRASTRUCTURE STACKS (v2.0.0)

â• â•

> **Dependencies:** Sections 0, 1 **Creates:** VPC, Aurora, DynamoDB, S3, KMS, WAF stacks

IMPORTANT: Type Imports

All CDK stacks should import types and configs from @radiant/shared:

```
// At the top of each stack file:
import {
  TierConfig,
  TierLevel,
  getTierConfig,
  TIER_CONFIGS,
  REGIONS,
  getMultiRegionDeployment,
} from '@radiant/shared';
```

DO NOT recreate the following files (they come from Section 0): - `lib/config/tiers.ts` - Import from @radiant/shared instead - `lib/config/regions.ts` - Import from @radiant/shared instead

DO create these CDK-specific files: - `lib/config/tags.ts` - CDK tagging utilities (shown in Section 1)

RADIANT v2.2.0 - Prompt 2: CDK Infrastructure Stacks

Prompt 2 of 9 | Target Size: ~35KB | Version: 3.7.0 | December 2024

OVERVIEW

This prompt creates the core AWS CDK infrastructure stacks:

1. **Foundation Stack** - SSM parameters, tagging, base configuration
2. **Networking Stack** - VPC, subnets, NAT, VPC endpoints
3. **Security Stack** - KMS keys, IAM roles, security groups, WAF
4. **Data Stack** - Aurora PostgreSQL, DynamoDB, ElastiCache
5. **Storage Stack** - S3 buckets, CloudFront distributions

All stacks are tier-aware and scale automatically based on the selected infrastructure tier (1-5).

INFRASTRUCTURE PACKAGE STRUCTURE

```
packages/infrastructure/
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ package.json
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ tsconfig.json
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ cdk.json
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ bin/
Ã¢â€â€   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ radiant.ts                    # CDK app entry point
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ lib/
Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ config/
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ tiers.ts                # Tier configurations
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ regions.ts              # Region configuration
Ã¢â€â€   Ã¢â€â€   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ tags.ts                 # Tagging utilities
Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ stacks/
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ foundation.stack.ts
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ networking.stack.ts
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ security.stack.ts
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ data.stack.ts
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ storage.stack.ts
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ ai.stack.ts              # Prompt 3
Ã¢â€â€   Ã¢â€â€   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ api.stack.ts             # Prompt 3
Ã¢â€â€   Ã¢â€â€   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ admin.stack.ts           # Prompt 3
```

```
Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ constructs/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ index.ts
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ aurora-global.construct.ts
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ vpc-endpoints.construct.ts
Ã¢â€ â€š       Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ waf.construct.ts
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ lambda/                          # Prompts 4-5
Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ [See Prompts 4-5]
Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ migrations/
    Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ [See Prompt 7]
```

PART 1: PACKAGE CONFIGURATION

packages/infrastructure/package.json

```json
{
  "name": "@radiant/infrastructure",
  "version": "2.2.0",
  "private": true,
  "scripts": {
    "build": "tsc",
    "watch": "tsc -w",
    "clean": "rm -rf dist cdk.out",
    "cdk": "cdk",
    "synth": "cdk synth",
    "deploy": "cdk deploy --all",
    "deploy:dev": "cdk deploy --all --context environment=dev --context tier=1",
    "deploy:staging": "cdk deploy --all --context environment=staging --context tier=2",
    "deploy:prod": "cdk deploy --all --context environment=prod --context tier=3",
    "diff": "cdk diff",
    "destroy": "cdk destroy --all",
    "test": "jest"
  },
  "dependencies": {
    "aws-cdk-lib": "^2.120.0",
    "constructs": "^10.3.0",
    "source-map-support": "^0.5.21"
  },
  "devDependencies": {
    "@types/node": "^20.10.0",
    "aws-cdk": "^2.120.0",
    "typescript": "^5.3.0",
    "jest": "^29.7.0",
    "@types/jest": "^29.5.11",
    "ts-jest": "^29.1.1"
  }
```

```
}
```

packages/infrastructure/tsconfig.json

2.2 CDK ENTRY POINT

> **NOTE:** The bin/radiant.ts file imports from @radiant/shared, not local config files.

```
  natGateways: 1,
  enableFlowLogs: true,
  flowLogsRetentionDays: 14,
},

aurora: {
  instanceClass: 'db.r6g.large',
  minCapacity: 1,
  maxCapacity: 4,
  enableGlobal: false,
  readerCount: 1,
  backupRetentionDays: 14,
  enablePerformanceInsights: true,
  performanceInsightsRetentionDays: 7,
  deletionProtection: true,
  enableIAMAuth: true,
},

dynamodb: {
  billingMode: 'PAY_PER_REQUEST',
  enablePointInTimeRecovery: true,
  enableContributorInsights: false,
},

elasticache: {
  enabled: true,
  nodeType: 'cache.t4g.micro',
  numCacheNodes: 1,
  enableMultiAz: false,
  enableAutoFailover: false,
  snapshotRetentionDays: 1,
},

lambda: {
  memoryMB: 1024,
  timeoutSeconds: 60,
},
```

```
litellm: {
  taskCount: 2,
  cpu: 512,
  memory: 1024,
  enableAutoScaling: true,
  minTasks: 1,
  maxTasks: 4,
},

sagemaker: {
  enabled: false,
},

s3: {
  intelligentTiering: true,
  lifecycleRules: true,
  replicationEnabled: false,
  versioning: true,
},

security: {
  enableWaf: true,
  enableShield: false,
  enableGuardDuty: true,
  enableInspector: false,
  enableSecurityHub: false,
  enableMacie: false,
  kmsKeyRotation: true,
},

features: {
  multiRegion: false,
  enhancedMonitoring: true,
  xrayTracing: true,
  detailedCloudWatchMetrics: false,
},

estimatedCost: {
  min: 300,
  max: 800,
  notes: 'Production-ready, single region',
},

},
// ============================================================
// TIER 3: GROWTH - Growing Production // ========================================
```

3: { tier: 3, name: ‘GROWTH’, description: ‘Growing production workloads’,

```
vpc: {
  maxAzs: 3,
  natGateways: 2,
  enableFlowLogs: true,
  flowLogsRetentionDays: 30,
},

aurora: {
  instanceClass: 'db.r6g.xlarge',
  minCapacity: 2,
  maxCapacity: 16,
  enableGlobal: false,
  readerCount: 2,
  backupRetentionDays: 30,
  enablePerformanceInsights: true,
  performanceInsightsRetentionDays: 31,
  deletionProtection: true,
  enableIAMAuth: true,
},

dynamodb: {
  billingMode: 'PAY_PER_REQUEST',
  enablePointInTimeRecovery: true,
  enableContributorInsights: true,
},

elasticache: {
  enabled: true,
  nodeType: 'cache.r6g.large',
  numCacheNodes: 2,
  enableMultiAz: true,
  enableAutoFailover: true,
  snapshotRetentionDays: 7,
},

lambda: {
  memoryMB: 2048,
  timeoutSeconds: 120,
  reservedConcurrency: 100,
},

litellm: {
  taskCount: 4,
  cpu: 1024,
```

```
  memory: 2048,
  enableAutoScaling: true,
  minTasks: 2,
  maxTasks: 8,
},

sagemaker: {
  enabled: true,
  defaultInstanceType: 'ml.g4dn.xlarge',
  enableMultiModel: false,
},

s3: {
  intelligentTiering: true,
  lifecycleRules: true,
  replicationEnabled: false,
  versioning: true,
},

security: {
  enableWaf: true,
  enableShield: false,
  enableGuardDuty: true,
  enableInspector: true,
  enableSecurityHub: true,
  enableMacie: false,
  kmsKeyRotation: true,
},

features: {
  multiRegion: false,
  enhancedMonitoring: true,
  xrayTracing: true,
  detailedCloudWatchMetrics: true,
},

estimatedCost: {
  min: 1500,
  max: 4000,
  notes: 'High availability, full monitoring',
},
},
// ============================================================
// TIER 4: SCALE - Large Production + Multi-Region // ============================
4: { tier: 4, name: 'SCALE', description: 'High-scale production with multi-
```

region',

```
vpc: {
  maxAzs: 3,
  natGateways: 3,
  enableFlowLogs: true,
  flowLogsRetentionDays: 90,
},

aurora: {
  instanceClass: 'db.r6g.2xlarge',
  minCapacity: 4,
  maxCapacity: 64,
  enableGlobal: true,
  readerCount: 3,
  backupRetentionDays: 35,
  enablePerformanceInsights: true,
  performanceInsightsRetentionDays: 93,
  deletionProtection: true,
  enableIAMAuth: true,
},

dynamodb: {
  billingMode: 'PAY_PER_REQUEST',
  enablePointInTimeRecovery: true,
  enableContributorInsights: true,
},

elasticache: {
  enabled: true,
  nodeType: 'cache.r6g.xlarge',
  numCacheNodes: 3,
  enableMultiAz: true,
  enableAutoFailover: true,
  snapshotRetentionDays: 14,
},

lambda: {
  memoryMB: 3072,
  timeoutSeconds: 300,
  reservedConcurrency: 500,
  provisionedConcurrency: 10,
},

litellm: {
  taskCount: 8,
```

```
  cpu: 2048,
  memory: 4096,
  enableAutoScaling: true,
  minTasks: 4,
  maxTasks: 16,
},

sagemaker: {
  enabled: true,
  defaultInstanceType: 'ml.g5.xlarge',
  enableMultiModel: true,
},

s3: {
  intelligentTiering: true,
  lifecycleRules: true,
  replicationEnabled: true,
  versioning: true,
},

security: {
  enableWaf: true,
  enableShield: true,
  enableGuardDuty: true,
  enableInspector: true,
  enableSecurityHub: true,
  enableMacie: true,
  kmsKeyRotation: true,
},

features: {
  multiRegion: true,
  enhancedMonitoring: true,
  xrayTracing: true,
  detailedCloudWatchMetrics: true,
},

estimatedCost: {
  min: 5000,
  max: 15000,
  notes: 'Multi-region, advanced security',
},

},
// ============================================================
// TIER 5: ENTERPRISE - Full Compliance // ========================================
```

5: { tier: 5, name: ‘ENTERPRISE’, description: ‘Enterprise-scale with full compliance’,

```
vpc: {
  maxAzs: 3,
  natGateways: 3,
  enableFlowLogs: true,
  flowLogsRetentionDays: 365,
},

aurora: {
  instanceClass: 'db.r6g.4xlarge',
  minCapacity: 8,
  maxCapacity: 128,
  enableGlobal: true,
  readerCount: 5,
  backupRetentionDays: 35,
  enablePerformanceInsights: true,
  performanceInsightsRetentionDays: 731,
  deletionProtection: true,
  enableIAMAuth: true,
},

dynamodb: {
  billingMode: 'PAY_PER_REQUEST',
  enablePointInTimeRecovery: true,
  enableContributorInsights: true,
},

elasticache: {
  enabled: true,
  nodeType: 'cache.r6g.2xlarge',
  numCacheNodes: 6,
  enableMultiAz: true,
  enableAutoFailover: true,
  snapshotRetentionDays: 35,
},

lambda: {
  memoryMB: 4096,
  timeoutSeconds: 900,
  reservedConcurrency: 1000,
  provisionedConcurrency: 50,
},

litellm: {
```

```
  taskCount: 16,
  cpu: 4096,
  memory: 8192,
  enableAutoScaling: true,
  minTasks: 8,
  maxTasks: 32,
},

sagemaker: {
  enabled: true,
  defaultInstanceType: 'ml.g5.2xlarge',
  enableMultiModel: true,
},

s3: {
  intelligentTiering: true,
  lifecycleRules: true,
  replicationEnabled: true,
  versioning: true,
},

security: {
  enableWaf: true,
  enableShield: true,
  enableGuardDuty: true,
  enableInspector: true,
  enableSecurityHub: true,
  enableMacie: true,
  kmsKeyRotation: true,
},

features: {
  multiRegion: true,
  enhancedMonitoring: true,
  xrayTracing: true,
  detailedCloudWatchMetrics: true,
},

estimatedCost: {
  min: 15000,
  max: 50000,
  notes: 'Full enterprise, HIPAA/SOC 2',
},
```

}, };

/** * Get tier configuration */ export function getTierConfig(tier: TierLevel):

TierConfig { const config = TIER_CONFIGS[tier]; if (!config) { throw new Error(`Invalid tier: ${tier}. Must be 1-5.`); } return config; }

/** * Validate tier for environment */ export function validateTierForEnvironment(tier: TierLevel, environment: string): void { if (environment === 'prod' && tier < 2) { console.warn('Ã¢Å¡Â Ã¯Â¸Â Warning: Tier 1 (SEED) is not recommended for production'); } if (environment === 'dev' && tier > 2) { console.warn('Ã¢Å¡Â Ã¯Â¸Â Warning: Tier 3+ may be expensive for development'); } }

packages/infrastructure/lib/config/regions.ts

```typescript
/**
 * AWS Region Configuration
 */

export interface RegionConfig {
  code: string;
  name: string;
  geography: 'us' | 'eu' | 'apac';
  isPrimary: boolean;
  supportsAllServices: boolean;
  latencyZone: number;  // 1 = lowest latency from US
}

export const REGIONS: Record<string, RegionConfig> = {
  'us-east-1': {
    code: 'us-east-1',
    name: 'US East (N. Virginia)',
    geography: 'us',
    isPrimary: true,
    supportsAllServices: true,
    latencyZone: 1,
  },
  'us-west-2': {
    code: 'us-west-2',
    name: 'US West (Oregon)',
    geography: 'us',
    isPrimary: false,
    supportsAllServices: true,
    latencyZone: 2,
  },
  'eu-west-1': {
    code: 'eu-west-1',
    name: 'Europe (Ireland)',
```

```
    geography: 'eu',
    isPrimary: false,
    supportsAllServices: true,
    latencyZone: 3,
  },
  'eu-central-1': {
    code: 'eu-central-1',
    name: 'Europe (Frankfurt)',
    geography: 'eu',
    isPrimary: false,
    supportsAllServices: true,
    latencyZone: 3,
  },
  'ap-northeast-1': {
    code: 'ap-northeast-1',
    name: 'Asia Pacific (Tokyo)',
    geography: 'apac',
    isPrimary: false,
    supportsAllServices: true,
    latencyZone: 4,
  },
  'ap-southeast-1': {
    code: 'ap-southeast-1',
    name: 'Asia Pacific (Singapore)',
    geography: 'apac',
    isPrimary: false,
    supportsAllServices: true,
    latencyZone: 4,
  },
};

/**
 * Default multi-region configuration
 * Maps geography to preferred region
 */
export const MULTI_REGION_CONFIG: Record<string, string> = {
  us: 'us-east-1',
  eu: 'eu-west-1',
  apac: 'ap-northeast-1',
};

/**
 * Get regions for multi-region deployment
 */
export function getMultiRegionDeployment(primaryRegion: string): string[] {
  const regions = new Set<string>([primaryRegion]);
```

```
  for (const region of Object.values(MULTI_REGION_CONFIG)) {
    regions.add(region);
  }

  return Array.from(regions);
}

/**
 * Get region configuration
 */
export function getRegionConfig(region: string): RegionConfig {
  const config = REGIONS[region];
  if (!config) {
    throw new Error(`Unsupported region: ${region}`);
  }
  return config;
}
```

packages/infrastructure/lib/config/tags.ts

```
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';

export interface RadiantTags {
  project?: string;
  appId: string;
  environment: string;
  tier: number;
  version?: string;
  owner?: string;
  costCenter?: string;
  compliance?: string;
}

/**
 * Apply standard RADIANT tags to a construct
 */
export function applyTags(scope: Construct, tags: RadiantTags): void {
  cdk.Tags.of(scope).add('Project', tags.project || 'RADIANT');
  cdk.Tags.of(scope).add('AppId', tags.appId);
  cdk.Tags.of(scope).add('Environment', tags.environment);
  cdk.Tags.of(scope).add('Tier', String(tags.tier));
  cdk.Tags.of(scope).add('Version', tags.version || '2.2.0');
  cdk.Tags.of(scope).add('ManagedBy', 'CDK');
```

```
  if (tags.owner) {
    cdk.Tags.of(scope).add('Owner', tags.owner);
  }
  if (tags.costCenter) {
    cdk.Tags.of(scope).add('CostCenter', tags.costCenter);
  }
  if (tags.compliance) {
    cdk.Tags.of(scope).add('Compliance', tags.compliance);
  }
}

/**
 * Get standard resource naming
 */
export function getResourceName(
  appId: string,
  environment: string,
  resourceType: string,
  suffix?: string
): string {
  const base = `${appId}-${environment}-${resourceType}`;
  return suffix ? `${base}-${suffix}` : base;
}
```

PART 4: FOUNDATION STACK

packages/infrastructure/lib/stacks/foundation.stack.ts

```
import * as cdk from 'aws-cdk-lib';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface FoundationStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
}

/**
 * Foundation Stack
```

```
 *
 * Creates base configuration and SSM parameters used by all other stacks.
 * This stack has no dependencies and is deployed first.
 */
export class FoundationStack extends cdk.Stack {
  public readonly tierParameter: ssm.StringParameter;
  public readonly configParameter: ssm.StringParameter;

  constructor(scope: Construct, id: string, props: FoundationStackProps) {
    super(scope, id, props);

    const prefix = `/radiant/${props.appId}/${props.environment}`;

    // ============================================================================
    // SSM PARAMETERS
    // ============================================================================

    // Tier level
    this.tierParameter = new ssm.StringParameter(this, 'TierParameter', {
      parameterName: `${prefix}/tier`,
      stringValue: String(props.tier),
      description: `RADIANT infrastructure tier for ${props.appName} ${props.environment}`,
      tier: ssm.ParameterTier.STANDARD,
    });

    // Full tier configuration (JSON)
    this.configParameter = new ssm.StringParameter(this, 'TierConfigParameter', {
      parameterName: `${prefix}/tier-config`,
      stringValue: JSON.stringify(props.tierConfig),
      description: `RADIANT tier configuration for ${props.appName} ${props.environment}`,
      tier: ssm.ParameterTier.ADVANCED,  // Allows larger values
    });

    // App metadata
    new ssm.StringParameter(this, 'AppIdParameter', {
      parameterName: `${prefix}/app-id`,
      stringValue: props.appId,
      description: 'Application identifier',
    });

    new ssm.StringParameter(this, 'AppNameParameter', {
      parameterName: `${prefix}/app-name`,
      stringValue: props.appName,
      description: 'Application display name',
    });
```

```typescript
new ssm.StringParameter(this, 'DomainParameter', {
  parameterName: `${prefix}/domain`,
  stringValue: props.domain,
  description: 'Base domain for the application',
});

new ssm.StringParameter(this, 'VersionParameter', {
  parameterName: `${prefix}/version`,
  stringValue: '2.2.0',
  description: 'RADIANT platform version',
});

new ssm.StringParameter(this, 'DeployedAtParameter', {
  parameterName: `${prefix}/deployed-at`,
  stringValue: new Date().toISOString(),
  description: 'Last deployment timestamp',
});

// Feature flags based on tier
new ssm.StringParameter(this, 'FeaturesParameter', {
  parameterName: `${prefix}/features`,
  stringValue: JSON.stringify({
    multiRegion: props.tierConfig.features.multiRegion,
    waf: props.tierConfig.security.enableWaf,
    guardDuty: props.tierConfig.security.enableGuardDuty,
    sagemaker: props.tierConfig.sagemaker.enabled,
    elasticache: props.tierConfig.elasticache.enabled,
    xray: props.tierConfig.features.xrayTracing,
  }),
  description: 'Enabled features for this deployment',
});

// ============================================================================
// TAGS
// ============================================================================

applyTags(this, {
  appId: props.appId,
  environment: props.environment,
  tier: props.tier,
});

// ============================================================================
// OUTPUTS
// ============================================================================
```

```
    new cdk.CfnOutput(this, 'TierName', {
      value: props.tierConfig.name,
      description: 'Infrastructure tier name',
      exportName: `${props.appId}-${props.environment}-tier-name`,
    });

    new cdk.CfnOutput(this, 'ParameterPrefix', {
      value: prefix,
      description: 'SSM parameter prefix',
      exportName: `${props.appId}-${props.environment}-param-prefix`,
    });
  }
}
```

PART 5: NETWORKING STACK

packages/infrastructure/lib/stacks/networking.stack.ts

```
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as logs from 'aws-cdk-lib/aws-logs';
import * as iam from 'aws-cdk-lib/aws-iam';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface NetworkingStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
}

/**
 * Networking Stack
 *
 * Creates VPC with public, private, and isolated subnets.
 * Configures NAT gateways, VPC endpoints, and flow logs.
 */
export class NetworkingStack extends cdk.Stack {
  public readonly vpc: ec2.Vpc;
  public readonly flowLogsGroup: logs.LogGroup;
```

```
constructor(scope: Construct, id: string, props: NetworkingStackProps) {
  super(scope, id, props);

  const { tierConfig } = props;

  // ============================================================================
  // VPC FLOW LOGS
  // ============================================================================

  this.flowLogsGroup = new logs.LogGroup(this, 'VpcFlowLogs', {
    logGroupName: `/radiant/${props.appId}/${props.environment}/vpc-flow-logs`,
    retention: this.getLogRetention(tierConfig.vpc.flowLogsRetentionDays),
    removalPolicy: props.environment === 'prod'
      ? cdk.RemovalPolicy.RETAIN
      : cdk.RemovalPolicy.DESTROY,
  });

  const flowLogsRole = new iam.Role(this, 'VpcFlowLogsRole', {
    assumedBy: new iam.ServicePrincipal('vpc-flow-logs.amazonaws.com'),
    description: 'Role for VPC Flow Logs',
  });

  this.flowLogsGroup.grantWrite(flowLogsRole);

  // ============================================================================
  // VPC
  // ============================================================================

  this.vpc = new ec2.Vpc(this, 'Vpc', {
    vpcName: `${props.appId}-${props.environment}-vpc`,

    // IP addressing
    ipAddresses: ec2.IpAddresses.cidr('10.0.0.0/16'),

    // Availability zones
    maxAzs: tierConfig.vpc.maxAzs,

    // NAT configuration
    natGateways: tierConfig.vpc.natGateways,
    natGatewayProvider: tierConfig.tier >= 3
      ? ec2.NatProvider.gateway()
      : ec2.NatProvider.gateway(),  // Could use NAT instances for tier 1

    // Subnet configuration
    subnetConfiguration: [
      {
```

```
      name: 'Public',
      subnetType: ec2.SubnetType.PUBLIC,
      cidrMask: 24,
      mapPublicIpOnLaunch: false,
    },
    {
      name: 'Private',
      subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS,
      cidrMask: 24,
    },
    {
      name: 'Isolated',
      subnetType: ec2.SubnetType.PRIVATE_ISOLATED,
      cidrMask: 24,
    },
  ],

  // Enable DNS
  enableDnsHostnames: true,
  enableDnsSupport: true,

  // Flow logs
  flowLogs: tierConfig.vpc.enableFlowLogs ? {
    flowLog: {
      destination: ec2.FlowLogDestination.toCloudWatchLogs(
        this.flowLogsGroup,
        flowLogsRole
      ),
      trafficType: ec2.FlowLogTrafficType.ALL,
    },
  } : undefined,
});

// ============================================================================
// VPC ENDPOINTS - GATEWAY (Free)
// ============================================================================

// S3 Gateway Endpoint (free)
this.vpc.addGatewayEndpoint('S3Endpoint', {
  service: ec2.GatewayVpcEndpointAwsService.S3,
  subnets: [
    { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
    { subnetType: ec2.SubnetType.PRIVATE_ISOLATED },
  ],
});
```

```
// DynamoDB Gateway Endpoint (free)
this.vpc.addGatewayEndpoint('DynamoDbEndpoint', {
  service: ec2.GatewayVpcEndpointAwsService.DYNAMODB,
  subnets: [
    { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
    { subnetType: ec2.SubnetType.PRIVATE_ISOLATED },
  ],
});

// ============================================================================
// VPC ENDPOINTS - INTERFACE (Tier 2+)
// ============================================================================

if (tierConfig.tier >= 2) {
  // Security group for VPC endpoints
  const endpointSecurityGroup = new ec2.SecurityGroup(this, 'EndpointSecurityGroup', {
    vpc: this.vpc,
    description: 'Security group for VPC Interface Endpoints',
    allowAllOutbound: false,
  });

  // Allow HTTPS from within VPC
  endpointSecurityGroup.addIngressRule(
    ec2.Peer.ipv4(this.vpc.vpcCidrBlock),
    ec2.Port.tcp(443),
    'Allow HTTPS from VPC'
  );

  // Secrets Manager
  this.vpc.addInterfaceEndpoint('SecretsManagerEndpoint', {
    service: ec2.InterfaceVpcEndpointAwsService.SECRETS_MANAGER,
    securityGroups: [endpointSecurityGroup],
    subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
    privateDnsEnabled: true,
  });

  // SSM
  this.vpc.addInterfaceEndpoint('SsmEndpoint', {
    service: ec2.InterfaceVpcEndpointAwsService.SSM,
    securityGroups: [endpointSecurityGroup],
    subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
    privateDnsEnabled: true,
  });

  // CloudWatch Logs
  this.vpc.addInterfaceEndpoint('CloudWatchLogsEndpoint', {
```

```
      service: ec2.InterfaceVpcEndpointAwsService.CLOUDWATCH_LOGS,
      securityGroups: [endpointSecurityGroup],
      subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
      privateDnsEnabled: true,
    });
  }

  // ============================================================================
  // VPC ENDPOINTS - ADDITIONAL (Tier 3+)
  // ============================================================================

  if (tierConfig.tier >= 3) {
    const endpointSecurityGroup = ec2.SecurityGroup.fromSecurityGroupId(
      this,
      'ExistingEndpointSg',
      this.vpc.vpcEndpoints[0]?.connections?.securityGroups[0]?.securityGroupId || '',
    );

    // ECR for container images
    this.vpc.addInterfaceEndpoint('EcrEndpoint', {
      service: ec2.InterfaceVpcEndpointAwsService.ECR,
      subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
      privateDnsEnabled: true,
    });

    this.vpc.addInterfaceEndpoint('EcrDockerEndpoint', {
      service: ec2.InterfaceVpcEndpointAwsService.ECR_DOCKER,
      subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
      privateDnsEnabled: true,
    });

    // SageMaker Runtime (if enabled)
    if (tierConfig.sagemaker.enabled) {
      this.vpc.addInterfaceEndpoint('SageMakerRuntimeEndpoint', {
        service: ec2.InterfaceVpcEndpointAwsService.SAGEMAKER_RUNTIME,
        subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
        privateDnsEnabled: true,
      });
    }

    // Lambda
    this.vpc.addInterfaceEndpoint('LambdaEndpoint', {
      service: ec2.InterfaceVpcEndpointAwsService.LAMBDA,
      subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
      privateDnsEnabled: true,
    });
```

```
    // STS
    this.vpc.addInterfaceEndpoint('StsEndpoint', {
      service: ec2.InterfaceVpcEndpointAwsService.STS,
      subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
      privateDnsEnabled: true,
    });

    // KMS
    this.vpc.addInterfaceEndpoint('KmsEndpoint', {
      service: ec2.InterfaceVpcEndpointAwsService.KMS,
      subnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
      privateDnsEnabled: true,
    });
  }

  // ============================================================================
  // TAGS
  // ============================================================================

  applyTags(this, {
    appId: props.appId,
    environment: props.environment,
    tier: props.tier,
  });

  // Tag subnets for easier identification
  for (const subnet of this.vpc.publicSubnets) {
    cdk.Tags.of(subnet).add('Name', `${props.appId}-${props.environment}-public-${subnet.a
    cdk.Tags.of(subnet).add('SubnetType', 'Public');
  }
  for (const subnet of this.vpc.privateSubnets) {
    cdk.Tags.of(subnet).add('Name', `${props.appId}-${props.environment}-private-${subnet
    cdk.Tags.of(subnet).add('SubnetType', 'Private');
  }
  for (const subnet of this.vpc.isolatedSubnets) {
    cdk.Tags.of(subnet).add('Name', `${props.appId}-${props.environment}-isolated-${subnet
    cdk.Tags.of(subnet).add('SubnetType', 'Isolated');
  }

  // ============================================================================
  // OUTPUTS
  // ============================================================================

  new cdk.CfnOutput(this, 'VpcId', {
    value: this.vpc.vpcId,
```

```
      description: 'VPC ID',
      exportName: `${props.appId}-${props.environment}-vpc-id`,
    });

    new cdk.CfnOutput(this, 'VpcCidr', {
      value: this.vpc.vpcCidrBlock,
      description: 'VPC CIDR block',
      exportName: `${props.appId}-${props.environment}-vpc-cidr`,
    });

    new cdk.CfnOutput(this, 'PrivateSubnets', {
      value: this.vpc.privateSubnets.map(s => s.subnetId).join(','),
      description: 'Private subnet IDs',
      exportName: `${props.appId}-${props.environment}-private-subnets`,
    });

    new cdk.CfnOutput(this, 'IsolatedSubnets', {
      value: this.vpc.isolatedSubnets.map(s => s.subnetId).join(','),
      description: 'Isolated subnet IDs',
      exportName: `${props.appId}-${props.environment}-isolated-subnets`,
    });
  }

  /**
   * Convert days to CloudWatch log retention enum
   */
  private getLogRetention(days: number): logs.RetentionDays {
    const retentionMap: Record<number, logs.RetentionDays> = {
      1: logs.RetentionDays.ONE_DAY,
      3: logs.RetentionDays.THREE_DAYS,
      5: logs.RetentionDays.FIVE_DAYS,
      7: logs.RetentionDays.ONE_WEEK,
      14: logs.RetentionDays.TWO_WEEKS,
      30: logs.RetentionDays.ONE_MONTH,
      60: logs.RetentionDays.TWO_MONTHS,
      90: logs.RetentionDays.THREE_MONTHS,
      120: logs.RetentionDays.FOUR_MONTHS,
      150: logs.RetentionDays.FIVE_MONTHS,
      180: logs.RetentionDays.SIX_MONTHS,
      365: logs.RetentionDays.ONE_YEAR,
      400: logs.RetentionDays.THIRTEEN_MONTHS,
      545: logs.RetentionDays.EIGHTEEN_MONTHS,
      731: logs.RetentionDays.TWO_YEARS,
      1827: logs.RetentionDays.FIVE_YEARS,
      3653: logs.RetentionDays.TEN_YEARS,
    };
```

```
    // Find closest match
    const sortedDays = Object.keys(retentionMap)
      .map(Number)
      .sort((a, b) => a - b);

    for (const d of sortedDays) {
      if (d >= days) {
        return retentionMap[d];
      }
    }

    return logs.RetentionDays.TEN_YEARS;
  }
}
```

PART 6: SECURITY STACK

packages/infrastructure/lib/stacks/security.stack.ts

```
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as kms from 'aws-cdk-lib/aws-kms';
import * as iam from 'aws-cdk-lib/aws-iam';
import * as wafv2 from 'aws-cdk-lib/aws-wafv2';
import * as guardduty from 'aws-cdk-lib/aws-guardduty';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface SecurityStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
  vpc: ec2.Vpc;
}

/**
 * Security Stack
 *
 * Creates KMS keys, IAM roles, security groups, and WAF rules.
 * Configures GuardDuty, Inspector, and other security services based on tier.
```

```
 */
export class SecurityStack extends cdk.Stack {
  // KMS Keys
  public readonly dataKey: kms.Key;
  public readonly mediaKey: kms.Key;
  public readonly logsKey: kms.Key;
  public readonly secretsKey: kms.Key;

  // Security Groups
  public readonly lambdaSecurityGroup: ec2.SecurityGroup;
  public readonly databaseSecurityGroup: ec2.SecurityGroup;
  public readonly cacheSecurityGroup: ec2.SecurityGroup;
  public readonly ecsSecurityGroup: ec2.SecurityGroup;
  public readonly albSecurityGroup: ec2.SecurityGroup;

  // WAF
  public readonly webAcl?: wafv2.CfnWebACL;

  // IAM Roles
  public readonly lambdaExecutionRole: iam.Role;

  constructor(scope: Construct, id: string, props: SecurityStackProps) {
    super(scope, id, props);

    const { tierConfig, vpc } = props;
    const resourcePrefix = `${props.appId}-${props.environment}`;

    // ============================================================================
    // KMS KEYS
    // ============================================================================

    // Data encryption key (Aurora, DynamoDB)
    this.dataKey = new kms.Key(this, 'DataKey', {
      alias: `alias/${resourcePrefix}-data`,
      description: `RADIANT data encryption key for ${props.appName} ${props.environment}`,
      enableKeyRotation: tierConfig.security.kmsKeyRotation,
      removalPolicy: props.environment === 'prod'
        ? cdk.RemovalPolicy.RETAIN
        : cdk.RemovalPolicy.DESTROY,
      pendingWindow: cdk.Duration.days(7),
    });

    // Media encryption key (S3)
    this.mediaKey = new kms.Key(this, 'MediaKey', {
      alias: `alias/${resourcePrefix}-media`,
      description: `RADIANT media encryption key for ${props.appName} ${props.environment}`,
```

```
  enableKeyRotation: tierConfig.security.kmsKeyRotation,
  removalPolicy: props.environment === 'prod'
    ? cdk.RemovalPolicy.RETAIN
    : cdk.RemovalPolicy.DESTROY,
  pendingWindow: cdk.Duration.days(7),
});

// Logs encryption key
this.logsKey = new kms.Key(this, 'LogsKey', {
  alias: `alias/${resourcePrefix}-logs`,
  description: `RADIANT logs encryption key for ${props.appName} ${props.environment}`,
  enableKeyRotation: tierConfig.security.kmsKeyRotation,
  removalPolicy: cdk.RemovalPolicy.DESTROY,
  pendingWindow: cdk.Duration.days(7),
});

// Allow CloudWatch Logs to use the key
this.logsKey.addToResourcePolicy(new iam.PolicyStatement({
  principals: [new iam.ServicePrincipal(`logs.${this.region}.amazonaws.com`)],
  actions: [
    'kms:Encrypt*',
    'kms:Decrypt*',
    'kms:ReEncrypt*',
    'kms:GenerateDataKey*',
    'kms:Describe*',
  ],
  resources: ['*'],
  conditions: {
    ArnLike: {
      'kms:EncryptionContext:aws:logs:arn': `arn:aws:logs:${this.region}:${this.account}
    },
  },
}));

// Secrets encryption key
this.secretsKey = new kms.Key(this, 'SecretsKey', {
  alias: `alias/${resourcePrefix}-secrets`,
  description: `RADIANT secrets encryption key for ${props.appName} ${props.environment}
  enableKeyRotation: tierConfig.security.kmsKeyRotation,
  removalPolicy: props.environment === 'prod'
    ? cdk.RemovalPolicy.RETAIN
    : cdk.RemovalPolicy.DESTROY,
  pendingWindow: cdk.Duration.days(7),
});

// ============================================================================
```

```
// SECURITY GROUPS
// ============================================================================

// ALB Security Group
this.albSecurityGroup = new ec2.SecurityGroup(this, 'AlbSecurityGroup', {
  vpc,
  securityGroupName: `${resourcePrefix}-alb-sg`,
  description: 'Security group for Application Load Balancer',
  allowAllOutbound: true,
});

this.albSecurityGroup.addIngressRule(
  ec2.Peer.anyIpv4(),
  ec2.Port.tcp(443),
  'Allow HTTPS from anywhere'
);

this.albSecurityGroup.addIngressRule(
  ec2.Peer.anyIpv4(),
  ec2.Port.tcp(80),
  'Allow HTTP for redirect'
);

// Lambda Security Group
this.lambdaSecurityGroup = new ec2.SecurityGroup(this, 'LambdaSecurityGroup', {
  vpc,
  securityGroupName: `${resourcePrefix}-lambda-sg`,
  description: 'Security group for Lambda functions',
  allowAllOutbound: true,
});

// ECS Security Group (for LiteLLM)
this.ecsSecurityGroup = new ec2.SecurityGroup(this, 'EcsSecurityGroup', {
  vpc,
  securityGroupName: `${resourcePrefix}-ecs-sg`,
  description: 'Security group for ECS Fargate tasks',
  allowAllOutbound: true,
});

this.ecsSecurityGroup.addIngressRule(
  this.albSecurityGroup,
  ec2.Port.tcp(4000),
  'Allow traffic from ALB to LiteLLM'
);

this.ecsSecurityGroup.addIngressRule(
```

```
    this.lambdaSecurityGroup,
    ec2.Port.tcp(4000),
    'Allow Lambda to call LiteLLM'
  );

  // Database Security Group
  this.databaseSecurityGroup = new ec2.SecurityGroup(this, 'DatabaseSecurityGroup', {
    vpc,
    securityGroupName: `${resourcePrefix}-db-sg`,
    description: 'Security group for Aurora PostgreSQL',
    allowAllOutbound: false,
  });

  this.databaseSecurityGroup.addIngressRule(
    this.lambdaSecurityGroup,
    ec2.Port.tcp(5432),
    'Allow Lambda to connect to Aurora'
  );

  this.databaseSecurityGroup.addIngressRule(
    this.ecsSecurityGroup,
    ec2.Port.tcp(5432),
    'Allow ECS to connect to Aurora'
  );

  // Cache Security Group
  this.cacheSecurityGroup = new ec2.SecurityGroup(this, 'CacheSecurityGroup', {
    vpc,
    securityGroupName: `${resourcePrefix}-cache-sg`,
    description: 'Security group for ElastiCache Redis',
    allowAllOutbound: false,
  });

  if (tierConfig.elasticache.enabled) {
    this.cacheSecurityGroup.addIngressRule(
      this.lambdaSecurityGroup,
      ec2.Port.tcp(6379),
      'Allow Lambda to connect to Redis'
    );

    this.cacheSecurityGroup.addIngressRule(
      this.ecsSecurityGroup,
      ec2.Port.tcp(6379),
      'Allow ECS to connect to Redis'
    );
  }
```

```
// ============================================================================
// IAM ROLES
// ============================================================================

// Lambda execution role
this.lambdaExecutionRole = new iam.Role(this, 'LambdaExecutionRole', {
  roleName: `${resourcePrefix}-lambda-execution`,
  assumedBy: new iam.ServicePrincipal('lambda.amazonaws.com'),
  description: 'Execution role for RADIANT Lambda functions',
  managedPolicies: [
    iam.ManagedPolicy.fromAwsManagedPolicyName('service-role/AWSLambdaVPCAccessExecutior
  ],
});

// Basic Lambda permissions
this.lambdaExecutionRole.addToPolicy(new iam.PolicyStatement({
  sid: 'CloudWatchLogs',
  actions: [
    'logs:CreateLogGroup',
    'logs:CreateLogStream',
    'logs:PutLogEvents',
  ],
  resources: [`arn:aws:logs:${this.region}:${this.account}:log-group:/aws/lambda/${resou
}));

// SSM Parameter access
this.lambdaExecutionRole.addToPolicy(new iam.PolicyStatement({
  sid: 'SSMParameters',
  actions: [
    'ssm:GetParameter',
    'ssm:GetParameters',
    'ssm:GetParametersByPath',
  ],
  resources: [`arn:aws:ssm:${this.region}:${this.account}:parameter/radiant/${props.appl
}));

// KMS access
this.lambdaExecutionRole.addToPolicy(new iam.PolicyStatement({
  sid: 'KMSAccess',
  actions: [
    'kms:Decrypt',
    'kms:GenerateDataKey',
  ],
  resources: [
    this.dataKey.keyArn,
```

```
    this.secretsKey.keyArn,
  ],
}));

// Secrets Manager access
this.lambdaExecutionRole.addToPolicy(new iam.PolicyStatement({
  sid: 'SecretsManager',
  actions: [
    'secretsmanager:GetSecretValue',
  ],
  resources: [`arn:aws:secretsmanager:${this.region}:${this.account}:secret:${resourcePr
}));

// X-Ray tracing (if enabled)
if (tierConfig.features.xrayTracing) {
  this.lambdaExecutionRole.addManagedPolicy(
    iam.ManagedPolicy.fromAwsManagedPolicyName('AWSXRayDaemonWriteAccess')
  );
}

// ============================================================================
// WAF (Tier 2+)
// ============================================================================

if (tierConfig.security.enableWaf) {
  this.webAcl = new wafv2.CfnWebACL(this, 'WebAcl', {
    name: `${resourcePrefix}-waf`,
    scope: 'REGIONAL',
    defaultAction: { allow: {} },
    visibilityConfig: {
      cloudWatchMetricsEnabled: true,
      metricName: `${resourcePrefix}-waf`,
      sampledRequestsEnabled: true,
    },
    rules: [
      // AWS Managed Rules - Common Rule Set
      {
        name: 'AWSManagedRulesCommonRuleSet',
        priority: 1,
        overrideAction: { none: {} },
        statement: {
          managedRuleGroupStatement: {
            vendorName: 'AWS',
            name: 'AWSManagedRulesCommonRuleSet',
          },
        },
```

```
    visibilityConfig: {
      cloudWatchMetricsEnabled: true,
      metricName: 'AWSManagedRulesCommonRuleSet',
      sampledRequestsEnabled: true,
    },
  },
  // AWS Managed Rules - Known Bad Inputs
  {
    name: 'AWSManagedRulesKnownBadInputsRuleSet',
    priority: 2,
    overrideAction: { none: {} },
    statement: {
      managedRuleGroupStatement: {
        vendorName: 'AWS',
        name: 'AWSManagedRulesKnownBadInputsRuleSet',
      },
    },
    visibilityConfig: {
      cloudWatchMetricsEnabled: true,
      metricName: 'AWSManagedRulesKnownBadInputsRuleSet',
      sampledRequestsEnabled: true,
    },
  },
  // AWS Managed Rules - SQL Injection
  {
    name: 'AWSManagedRulesSQLiRuleSet',
    priority: 3,
    overrideAction: { none: {} },
    statement: {
      managedRuleGroupStatement: {
        vendorName: 'AWS',
        name: 'AWSManagedRulesSQLiRuleSet',
      },
    },
    visibilityConfig: {
      cloudWatchMetricsEnabled: true,
      metricName: 'AWSManagedRulesSQLiRuleSet',
      sampledRequestsEnabled: true,
    },
  },
  // Rate limiting
  {
    name: 'RateLimitRule',
    priority: 10,
    action: { block: {} },
    statement: {
```

```
          rateBasedStatement: {
            limit: tierConfig.tier >= 4 ? 5000 : 2000,
            aggregateKeyType: 'IP',
          },
        },
        visibilityConfig: {
          cloudWatchMetricsEnabled: true,
          metricName: 'RateLimitRule',
          sampledRequestsEnabled: true,
        },
      },
    ],
  });
}

// ============================================================================
// GUARDDUTY (Tier 2+)
// ============================================================================

if (tierConfig.security.enableGuardDuty) {
  // Note: GuardDuty detector is typically enabled once per account/region
  // This creates it if it doesn't exist
  new guardduty.CfnDetector(this, 'GuardDutyDetector', {
    enable: true,
    findingPublishingFrequency: 'FIFTEEN_MINUTES',
    dataSources: {
      s3Logs: { enable: true },
      kubernetes: {
        auditLogs: { enable: false },
      },
      malwareProtection: {
        scanEc2InstanceWithFindings: {
          ebsVolumes: true,
        },
      },
    },
  });
}

// ============================================================================
// TAGS
// ============================================================================

applyTags(this, {
  appId: props.appId,
  environment: props.environment,
```

```
      tier: props.tier,
    });

    // ============================================================================
    // OUTPUTS
    // ============================================================================

    new cdk.CfnOutput(this, 'DataKeyArn', {
      value: this.dataKey.keyArn,
      description: 'Data encryption key ARN',
      exportName: `${resourcePrefix}-data-key-arn`,
    });

    new cdk.CfnOutput(this, 'MediaKeyArn', {
      value: this.mediaKey.keyArn,
      description: 'Media encryption key ARN',
      exportName: `${resourcePrefix}-media-key-arn`,
    });

    new cdk.CfnOutput(this, 'LambdaSecurityGroupId', {
      value: this.lambdaSecurityGroup.securityGroupId,
      description: 'Lambda security group ID',
      exportName: `${resourcePrefix}-lambda-sg-id`,
    });

    new cdk.CfnOutput(this, 'DatabaseSecurityGroupId', {
      value: this.databaseSecurityGroup.securityGroupId,
      description: 'Database security group ID',
      exportName: `${resourcePrefix}-db-sg-id`,
    });

    if (this.webAcl) {
      new cdk.CfnOutput(this, 'WebAclArn', {
        value: this.webAcl.attrArn,
        description: 'WAF Web ACL ARN',
        exportName: `${resourcePrefix}-waf-arn`,
      });
    }
  }
}
```

PART 7: DATA STACK

packages/infrastructure/lib/stacks/data.stack.ts

```typescript
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as rds from 'aws-cdk-lib/aws-rds';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as elasticache from 'aws-cdk-lib/aws-elasticache';
import * as kms from 'aws-cdk-lib/aws-kms';
import * as secretsmanager from 'aws-cdk-lib/aws-secretsmanager';
import * as logs from 'aws-cdk-lib/aws-logs';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface DataStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
  vpc: ec2.Vpc;
  databaseSecurityGroup: ec2.SecurityGroup;
  lambdaSecurityGroup: ec2.SecurityGroup;
  dataKey: kms.Key;
  isPrimary: boolean;
  enableGlobal: boolean;
  globalClusterIdentifier?: string;
}

/**
 * Data Stack
 *
 * Creates Aurora PostgreSQL Serverless v2, DynamoDB tables, and ElastiCache.
 * Supports Aurora Global Database for multi-region deployments.
 */
export class DataStack extends cdk.Stack {
  public readonly cluster: rds.DatabaseCluster;
  public readonly dbSecret: secretsmanager.ISecret;
  public readonly globalClusterIdentifier?: string;

  // DynamoDB Tables
  public readonly usageTable: dynamodb.Table;
  public readonly sessionsTable: dynamodb.Table;
```

```
public readonly cacheTable: dynamodb.Table;

// ElastiCache
public readonly redisCluster?: elasticache.CfnCacheCluster | elasticache.CfnReplicationGro
public readonly redisEndpoint?: string;

constructor(scope: Construct, id: string, props: DataStackProps) {
  super(scope, id, props);

  const { tierConfig, vpc } = props;
  const resourcePrefix = `${props.appId}-${props.environment}`;

  // ========================================================================
  // AURORA POSTGRESQL SERVERLESS V2
  // ========================================================================

  // Database credentials
  const dbCredentials = rds.Credentials.fromGeneratedSecret('radiant_admin', {
    secretName: `${resourcePrefix}/aurora/credentials`,
    encryptionKey: props.dataKey,
  });

  this.dbSecret = dbCredentials.secret!;

  // Subnet group for Aurora
  const subnetGroup = new rds.SubnetGroup(this, 'AuroraSubnetGroup', {
    vpc,
    description: `Subnet group for ${props.appName} Aurora cluster`,
    vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_ISOLATED },
    subnetGroupName: `${resourcePrefix}-aurora-subnet-group`,
  });

  // Parameter group with optimized settings
  const parameterGroup = new rds.ParameterGroup(this, 'AuroraParameterGroup', {
    engine: rds.DatabaseClusterEngine.auroraPostgres({
      version: rds.AuroraPostgresEngineVersion.VER_15_4,
    }),
    description: `RADIANT parameter group for ${props.appName}`,
    parameters: {
      // Security
      'rds.force_ssl': '1',

      // Logging
      'log_statement': 'ddl',
      'log_min_duration_statement': '1000',
      'log_connections': '1',
```

```
    'log_disconnections': '1',

    // Performance
    'shared_preload_libraries': 'pg_stat_statements,auto_explain',
    'pg_stat_statements.track': 'all',
    'auto_explain.log_min_duration': '3000',

    // Memory (will be adjusted by Serverless v2)
    'work_mem': '65536',
    'maintenance_work_mem': '524288',

    // WAL
    'wal_level': 'logical',
    'max_wal_senders': '10',

    // Row-Level Security
    'row_security': 'on',
  },
});

// CloudWatch Log exports
const cloudwatchLogsExports = tierConfig.tier >= 2
  ? ['postgresql', 'upgrade']
  : ['postgresql'];

// Determine if we're creating primary or secondary cluster
if (props.isPrimary) {
  // Primary cluster
  this.cluster = new rds.DatabaseCluster(this, 'AuroraCluster', {
    engine: rds.DatabaseClusterEngine.auroraPostgres({
      version: rds.AuroraPostgresEngineVersion.VER_15_4,
    }),

    clusterIdentifier: `${resourcePrefix}-aurora`,
    defaultDatabaseName: 'radiant',

    credentials: dbCredentials,

    // Serverless v2 configuration
    serverlessV2MinCapacity: tierConfig.aurora.minCapacity,
    serverlessV2MaxCapacity: tierConfig.aurora.maxCapacity,

    // Writer instance
    writer: rds.ClusterInstance.serverlessV2('Writer', {
      publiclyAccessible: false,
      enablePerformanceInsights: tierConfig.aurora.enablePerformanceInsights,
```

```
      performanceInsightRetention: tierConfig.aurora.enablePerformanceInsights
        ? this.getPerformanceInsightRetention(tierConfig.aurora.performanceInsightsReter
        : undefined,
      performanceInsightEncryptionKey: tierConfig.aurora.enablePerformanceInsights
        ? props.dataKey
        : undefined,
    }),

    // Reader instances
    readers: this.createReaderInstances(tierConfig),

    vpc,
    vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_ISOLATED },
    subnetGroup,
    securityGroups: [props.databaseSecurityGroup],

    parameterGroup,

    // Encryption
    storageEncrypted: true,
    storageEncryptionKey: props.dataKey,

    // Backup
    backup: {
      retention: cdk.Duration.days(tierConfig.aurora.backupRetentionDays),
      preferredWindow: '03:00-04:00',
    },

    // Maintenance
    preferredMaintenanceWindow: 'sun:04:00-sun:05:00',

    // High availability
    deletionProtection: tierConfig.aurora.deletionProtection,

    // Logging
    cloudwatchLogsExports,
    cloudwatchLogsRetention: logs.RetentionDays.ONE_MONTH,

    // IAM authentication
    iamAuthentication: tierConfig.aurora.enableIAMAuth,

    // Removal policy
    removalPolicy: props.environment === 'prod'
      ? cdk.RemovalPolicy.RETAIN
      : cdk.RemovalPolicy.DESTROY,
  });
```

```
    // Store global cluster identifier for secondary regions
    if (props.enableGlobal) {
      this.globalClusterIdentifier = `${resourcePrefix}-global`;
      // Note: Aurora Global Database requires additional configuration
      // that would be handled in a separate construct
    }
  } else {
    // Secondary cluster (Aurora Global Database reader)
    // This requires the global cluster to exist
    if (!props.globalClusterIdentifier) {
      throw new Error('globalClusterIdentifier required for secondary clusters');
    }

    this.cluster = new rds.DatabaseCluster(this, 'AuroraCluster', {
      engine: rds.DatabaseClusterEngine.auroraPostgres({
        version: rds.AuroraPostgresEngineVersion.VER_15_4,
      }),

      clusterIdentifier: `${resourcePrefix}-aurora-${this.region}`,

      // Serverless v2 for readers
      serverlessV2MinCapacity: tierConfig.aurora.minCapacity,
      serverlessV2MaxCapacity: tierConfig.aurora.maxCapacity,

      writer: rds.ClusterInstance.serverlessV2('Reader', {
        publiclyAccessible: false,
      }),

      vpc,
      vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_ISOLATED },
      subnetGroup,
      securityGroups: [props.databaseSecurityGroup],

      storageEncrypted: true,
      storageEncryptionKey: props.dataKey,

      removalPolicy: cdk.RemovalPolicy.DESTROY,
    });
  }

  // ============================================================================
  // DYNAMODB TABLES
  // ============================================================================

  const dynamodbConfig = {
```

```
  billingMode: tierConfig.dynamodb.billingMode === 'PAY_PER_REQUEST'
    ? dynamodb.BillingMode.PAY_PER_REQUEST
    : dynamodb.BillingMode.PROVISIONED,
  pointInTimeRecovery: tierConfig.dynamodb.enablePointInTimeRecovery,
  contributorInsightsEnabled: tierConfig.dynamodb.enableContributorInsights,
  encryption: dynamodb.TableEncryption.CUSTOMER_MANAGED,
  encryptionKey: props.dataKey,
};

// Usage/Metering Table
this.usageTable = new dynamodb.Table(this, 'UsageTable', {
  tableName: `${resourcePrefix}-usage`,
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING }, // tenantId
  sortKey: { name: 'sk', type: dynamodb.AttributeType.STRING },      // timestamp
  ...dynamodbConfig,
  timeToLiveAttribute: 'ttl',
  stream: dynamodb.StreamViewType.NEW_AND_OLD_IMAGES,
  removalPolicy: props.environment === 'prod'
    ? cdk.RemovalPolicy.RETAIN
    : cdk.RemovalPolicy.DESTROY,
});

// GSI for querying by model/provider
this.usageTable.addGlobalSecondaryIndex({
  indexName: 'gsi1',
  partitionKey: { name: 'gsi1pk', type: dynamodb.AttributeType.STRING }, // modelId
  sortKey: { name: 'gsi1sk', type: dynamodb.AttributeType.STRING },      // timestamp
  projectionType: dynamodb.ProjectionType.ALL,
});

// GSI for querying by period
this.usageTable.addGlobalSecondaryIndex({
  indexName: 'gsi2',
  partitionKey: { name: 'gsi2pk', type: dynamodb.AttributeType.STRING }, // period (YYY
  sortKey: { name: 'gsi2sk', type: dynamodb.AttributeType.STRING },      // tenantId#ti
  projectionType: dynamodb.ProjectionType.ALL,
});

// Sessions Table
this.sessionsTable = new dynamodb.Table(this, 'SessionsTable', {
  tableName: `${resourcePrefix}-sessions`,
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING }, // sessionId
  ...dynamodbConfig,
  timeToLiveAttribute: 'ttl',
  removalPolicy: cdk.RemovalPolicy.DESTROY,
});
```

```
// GSI for user sessions
this.sessionsTable.addGlobalSecondaryIndex({
  indexName: 'gsi1',
  partitionKey: { name: 'userId', type: dynamodb.AttributeType.STRING },
  sortKey: { name: 'createdAt', type: dynamodb.AttributeType.STRING },
  projectionType: dynamodb.ProjectionType.ALL,
});

// Cache Table (for API response caching)
this.cacheTable = new dynamodb.Table(this, 'CacheTable', {
  tableName: `${resourcePrefix}-cache`,
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING }, // cacheKey
  ...dynamodbConfig,
  timeToLiveAttribute: 'ttl',
  removalPolicy: cdk.RemovalPolicy.DESTROY,
});

// ============================================================================
// ELASTICACHE REDIS (Tier 2+)
// ============================================================================

if (tierConfig.elasticache.enabled) {
  // Subnet group for ElastiCache
  const cacheSubnetGroup = new elasticache.CfnSubnetGroup(this, 'CacheSubnetGroup', {
    subnetIds: vpc.privateSubnets.map(subnet => subnet.subnetId),
    description: `Subnet group for ${props.appName} ElastiCache`,
    cacheSubnetGroupName: `${resourcePrefix}-cache-subnet-group`,
  });

  // Get the cache security group from security stack
  // Note: This is passed in from the security stack
  const cacheSecurityGroupId = cdk.Fn.importValue(
    `${props.appId}-${props.environment}-cache-sg-id`
  );

  if (tierConfig.elasticache.enableMultiAz && tierConfig.elasticache.numCacheNodes! >= 2
    // Replication Group (Multi-AZ)
    const replicationGroup = new elasticache.CfnReplicationGroup(this, 'RedisReplicatio
      replicationGroupDescription: `RADIANT Redis for ${props.appName}`,
      replicationGroupId: `${resourcePrefix}-redis`,

      engine: 'redis',
      engineVersion: '7.0',

      cacheNodeType: tierConfig.elasticache.nodeType,
```

```
      numCacheClusters: tierConfig.elasticache.numCacheNodes,

      automaticFailoverEnabled: tierConfig.elasticache.enableAutoFailover,
      multiAzEnabled: tierConfig.elasticache.enableMultiAz,

      cacheSubnetGroupName: cacheSubnetGroup.cacheSubnetGroupName,
      securityGroupIds: [cacheSecurityGroupId],

      atRestEncryptionEnabled: true,
      transitEncryptionEnabled: true,

      snapshotRetentionLimit: tierConfig.elasticache.snapshotRetentionDays,
      snapshotWindow: '05:00-06:00',
      preferredMaintenanceWindow: 'sun:06:00-sun:07:00',

      autoMinorVersionUpgrade: true,
    });

    replicationGroup.addDependency(cacheSubnetGroup);

    this.redisCluster = replicationGroup;
    this.redisEndpoint = replicationGroup.attrPrimaryEndPointAddress;
  } else {
    // Single node (Tier 2)
    const cacheCluster = new elasticache.CfnCacheCluster(this, 'RedisCluster', {
      clusterName: `${resourcePrefix}-redis`,

      engine: 'redis',
      engineVersion: '7.0',

      cacheNodeType: tierConfig.elasticache.nodeType,
      numCacheNodes: 1,

      cacheSubnetGroupName: cacheSubnetGroup.cacheSubnetGroupName,
      vpcSecurityGroupIds: [cacheSecurityGroupId],

      snapshotRetentionLimit: tierConfig.elasticache.snapshotRetentionDays,
      snapshotWindow: '05:00-06:00',
      preferredMaintenanceWindow: 'sun:06:00-sun:07:00',

      autoMinorVersionUpgrade: true,
    });

    cacheCluster.addDependency(cacheSubnetGroup);

    this.redisCluster = cacheCluster;
```

```
    this.redisEndpoint = cacheCluster.attrRedisEndpointAddress;
  }
}

// ============================================================================
// TAGS
// ============================================================================

applyTags(this, {
  appId: props.appId,
  environment: props.environment,
  tier: props.tier,
});

// ============================================================================
// OUTPUTS
// ============================================================================

new cdk.CfnOutput(this, 'ClusterEndpoint', {
  value: this.cluster.clusterEndpoint.hostname,
  description: 'Aurora cluster endpoint',
  exportName: `${resourcePrefix}-aurora-endpoint`,
});

new cdk.CfnOutput(this, 'ClusterReaderEndpoint', {
  value: this.cluster.clusterReadEndpoint.hostname,
  description: 'Aurora cluster reader endpoint',
  exportName: `${resourcePrefix}-aurora-reader-endpoint`,
});

new cdk.CfnOutput(this, 'DbSecretArn', {
  value: this.dbSecret.secretArn,
  description: 'Database credentials secret ARN',
  exportName: `${resourcePrefix}-db-secret-arn`,
});

new cdk.CfnOutput(this, 'UsageTableName', {
  value: this.usageTable.tableName,
  description: 'Usage DynamoDB table name',
  exportName: `${resourcePrefix}-usage-table`,
});

new cdk.CfnOutput(this, 'SessionsTableName', {
  value: this.sessionsTable.tableName,
  description: 'Sessions DynamoDB table name',
  exportName: `${resourcePrefix}-sessions-table`,
```

```
    });

    if (this.redisEndpoint) {
      new cdk.CfnOutput(this, 'RedisEndpoint', {
        value: this.redisEndpoint,
        description: 'ElastiCache Redis endpoint',
        exportName: `${resourcePrefix}-redis-endpoint`,
      });
    }
  }

  /**
   * Create reader instances based on tier configuration
   */
  private createReaderInstances(tierConfig: TierConfig): rds.IClusterInstance[] {
    const readers: rds.IClusterInstance[] = [];

    for (let i = 0; i < tierConfig.aurora.readerCount; i++) {
      readers.push(
        rds.ClusterInstance.serverlessV2(`Reader${i + 1}`, {
          publiclyAccessible: false,
          enablePerformanceInsights: tierConfig.aurora.enablePerformanceInsights,
          performanceInsightRetention: tierConfig.aurora.enablePerformanceInsights
            ? this.getPerformanceInsightRetention(tierConfig.aurora.performanceInsightsReten
            : undefined,
        })
      );
    }

    return readers;
  }

  /**
   * Convert days to Performance Insights retention enum
   */
  private getPerformanceInsightRetention(days: number): rds.PerformanceInsightRetention {
    if (days <= 7) return rds.PerformanceInsightRetention.DEFAULT;
    if (days <= 31) return rds.PerformanceInsightRetention.MONTHS_1;
    if (days <= 93) return rds.PerformanceInsightRetention.MONTHS_3;
    if (days <= 186) return rds.PerformanceInsightRetention.MONTHS_6;
    if (days <= 372) return rds.PerformanceInsightRetention.MONTHS_12;
    return rds.PerformanceInsightRetention.MONTHS_24;
  }
}
```

PART 8: STORAGE STACK

packages/infrastructure/lib/stacks/storage.stack.ts

```typescript
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as s3 from 'aws-cdk-lib/aws-s3';
import * as cloudfront from 'aws-cdk-lib/aws-cloudfront';
import * as origins from 'aws-cdk-lib/aws-cloudfront-origins';
import * as kms from 'aws-cdk-lib/aws-kms';
import * as iam from 'aws-cdk-lib/aws-iam';
import * as route53 from 'aws-cdk-lib/aws-route53';
import * as acm from 'aws-cdk-lib/aws-certificatemanager';
import * as route53targets from 'aws-cdk-lib/aws-route53-targets';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface StorageStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
  vpc: ec2.Vpc;
  mediaKey: kms.Key;
  hostedZoneId?: string;
  enableReplication?: boolean;
}

/**
 * Storage Stack
 *
 * Creates S3 buckets for media, assets, and logs.
 * Configures CloudFront distributions for global content delivery.
 */
export class StorageStack extends cdk.Stack {
  // S3 Buckets
  public readonly mediaBucket: s3.Bucket;
  public readonly assetsBucket: s3.Bucket;
  public readonly logsBucket: s3.Bucket;

  // CloudFront
  public readonly mediaDistribution: cloudfront.Distribution;
  public readonly assetsDistribution: cloudfront.Distribution;
```

```
// URLs
public readonly mediaUrl: string;
public readonly assetsUrl: string;

constructor(scope: Construct, id: string, props: StorageStackProps) {
  super(scope, id, props);

  const { tierConfig, vpc } = props;
  const resourcePrefix = `${props.appId}-${props.environment}`;

  // ============================================================================
  // LOGS BUCKET (for access logs)
  // ============================================================================

  this.logsBucket = new s3.Bucket(this, 'LogsBucket', {
    bucketName: `${resourcePrefix}-logs-${this.account}-${this.region}`,

    encryption: s3.BucketEncryption.S3_MANAGED,

    blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,

    enforceSSL: true,

    lifecycleRules: [
      {
        id: 'TransitionToIA',
        transitions: [
          {
            storageClass: s3.StorageClass.INFREQUENT_ACCESS,
            transitionAfter: cdk.Duration.days(30),
          },
          {
            storageClass: s3.StorageClass.GLACIER,
            transitionAfter: cdk.Duration.days(90),
          },
        ],
        expiration: cdk.Duration.days(365),
      },
    ],

    removalPolicy: props.environment === 'prod'
      ? cdk.RemovalPolicy.RETAIN
      : cdk.RemovalPolicy.DESTROY,

    autoDeleteObjects: props.environment !== 'prod',
```

```
});

// ============================================================================
// MEDIA BUCKET (user uploads, AI-generated content)
// ============================================================================

this.mediaBucket = new s3.Bucket(this, 'MediaBucket', {
  bucketName: `${resourcePrefix}-media-${this.account}-${this.region}`,

  // Encryption with KMS
  encryption: s3.BucketEncryption.KMS,
  encryptionKey: props.mediaKey,
  bucketKeyEnabled: true,

  // Security
  blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
  enforceSSL: true,

  // Versioning
  versioned: tierConfig.s3.versioning,

  // CORS for direct uploads
  cors: [
    {
      allowedMethods: [
        s3.HttpMethods.GET,
        s3.HttpMethods.PUT,
        s3.HttpMethods.POST,
      ],
      allowedOrigins: [
        `https://${props.domain}`,
        `https://*.${props.domain}`,
        'http://localhost:*',
      ],
      allowedHeaders: ['*'],
      exposedHeaders: ['ETag'],
      maxAge: 3600,
    },
  ],

  // Lifecycle rules
  lifecycleRules: tierConfig.s3.lifecycleRules ? [
    {
      id: 'IntelligentTiering',
      enabled: tierConfig.s3.intelligentTiering,
      transitions: tierConfig.s3.intelligentTiering ? [
```

```
        {
          storageClass: s3.StorageClass.INTELLIGENT_TIERING,
          transitionAfter: cdk.Duration.days(0),
        },
      ] : [],
    },
    {
      id: 'CleanupIncompleteUploads',
      abortIncompleteMultipartUploadAfter: cdk.Duration.days(7),
    },
    {
      id: 'ExpireOldVersions',
      noncurrentVersionExpiration: cdk.Duration.days(90),
      enabled: tierConfig.s3.versioning,
    },
  ] : undefined,

  // Server access logging
  serverAccessLogsBucket: this.logsBucket,
  serverAccessLogsPrefix: 'media/',

  // Removal policy
  removalPolicy: props.environment === 'prod'
    ? cdk.RemovalPolicy.RETAIN
    : cdk.RemovalPolicy.DESTROY,

  autoDeleteObjects: props.environment !== 'prod',
});

// ============================================================================
// ASSETS BUCKET (static assets, compiled frontend)
// ============================================================================

this.assetsBucket = new s3.Bucket(this, 'AssetsBucket', {
  bucketName: `${resourcePrefix}-assets-${this.account}-${this.region}`,

  // S3-managed encryption for static assets
  encryption: s3.BucketEncryption.S3_MANAGED,

  blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
  enforceSSL: true,

  lifecycleRules: [
    {
      id: 'CleanupOldAssets',
      prefix: 'old/',
```

```
      expiration: cdk.Duration.days(30),
    },
  ],

  serverAccessLogsBucket: this.logsBucket,
  serverAccessLogsPrefix: 'assets/',

  removalPolicy: cdk.RemovalPolicy.DESTROY,
  autoDeleteObjects: true,
});

// ============================================================================
// CLOUDFRONT - MEDIA DISTRIBUTION
// ============================================================================

// Origin Access Identity for S3
const mediaOai = new cloudfront.OriginAccessIdentity(this, 'MediaOai', {
  comment: `OAI for ${props.appName} media bucket`,
});

this.mediaBucket.grantRead(mediaOai);

// Response headers policy
const responseHeadersPolicy = new cloudfront.ResponseHeadersPolicy(this, 'MediaResponseH
  responseHeadersPolicyName: `${resourcePrefix}-media-headers`,
  securityHeadersBehavior: {
    contentSecurityPolicy: {
      contentSecurityPolicy: "default-src 'self'",
      override: false,
    },
    contentTypeOptions: {
      override: true,
    },
    frameOptions: {
      frameOption: cloudfront.HeadersFrameOption.DENY,
      override: true,
    },
    referrerPolicy: {
      referrerPolicy: cloudfront.HeadersReferrerPolicy.STRICT_ORIGIN_WHEN_CROSS_ORIGIN,
      override: true,
    },
    strictTransportSecurity: {
      accessControlMaxAge: cdk.Duration.days(365),
      includeSubdomains: true,
      preload: true,
      override: true,
```

```
    },
    xssProtection: {
      protection: true,
      modeBlock: true,
      override: true,
    },
  },
  corsBehavior: {
    accessControlAllowCredentials: false,
    accessControlAllowHeaders: ['*'],
    accessControlAllowMethods: ['GET', 'HEAD'],
    accessControlAllowOrigins: [
      `https://${props.domain}`,
      `https://*.${props.domain}`,
    ],
    accessControlMaxAge: cdk.Duration.days(1),
    originOverride: true,
  },
});

// Cache policy for media
const mediaCachePolicy = new cloudfront.CachePolicy(this, 'MediaCachePolicy', {
  cachePolicyName: `${resourcePrefix}-media-cache`,
  defaultTtl: cdk.Duration.days(1),
  minTtl: cdk.Duration.seconds(0),
  maxTtl: cdk.Duration.days(365),
  enableAcceptEncodingGzip: true,
  enableAcceptEncodingBrotli: true,
  queryStringBehavior: cloudfront.CacheQueryStringBehavior.allowList('v', 'w', 'h', 'q')
});

// Media CloudFront distribution
this.mediaDistribution = new cloudfront.Distribution(this, 'MediaDistribution', {
  comment: `${props.appName} Media Distribution`,

  defaultBehavior: {
    origin: new origins.S3Origin(this.mediaBucket, {
      originAccessIdentity: mediaOai,
    }),
    viewerProtocolPolicy: cloudfront.ViewerProtocolPolicy.REDIRECT_TO_HTTPS,
    allowedMethods: cloudfront.AllowedMethods.ALLOW_GET_HEAD,
    cachedMethods: cloudfront.CachedMethods.CACHE_GET_HEAD,
    cachePolicy: mediaCachePolicy,
    responseHeadersPolicy,
    compress: true,
  },
```

```
// Additional behaviors for specific paths
additionalBehaviors: {
  '/uploads/*': {
    origin: new origins.S3Origin(this.mediaBucket, {
      originAccessIdentity: mediaOai,
    }),
    viewerProtocolPolicy: cloudfront.ViewerProtocolPolicy.REDIRECT_TO_HTTPS,
    allowedMethods: cloudfront.AllowedMethods.ALLOW_GET_HEAD,
    cachePolicy: cloudfront.CachePolicy.CACHING_OPTIMIZED,
    responseHeadersPolicy,
  },
},

// Error pages
errorResponses: [
  {
    httpStatus: 403,
    responseHttpStatus: 404,
    responsePagePath: '/404.html',
    ttl: cdk.Duration.minutes(5),
  },
  {
    httpStatus: 404,
    responseHttpStatus: 404,
    responsePagePath: '/404.html',
    ttl: cdk.Duration.minutes(5),
  },
],

// Geo restrictions (if needed for compliance)
geoRestriction: cloudfront.GeoRestriction.allowlist('US', 'CA', 'GB', 'DE', 'FR', 'JP'

// Price class based on tier
priceClass: tierConfig.tier >= 4
  ? cloudfront.PriceClass.PRICE_CLASS_ALL
  : tierConfig.tier >= 2
    ? cloudfront.PriceClass.PRICE_CLASS_100
    : cloudfront.PriceClass.PRICE_CLASS_100,

// HTTP/2 and HTTP/3
httpVersion: cloudfront.HttpVersion.HTTP2_AND_3,

// Logging
enableLogging: true,
logBucket: this.logsBucket,
```

```
  logFilePrefix: 'cloudfront/media/',

  // Minimum protocol version
  minimumProtocolVersion: cloudfront.SecurityPolicyProtocol.TLS_V1_2_2021,
});

this.mediaUrl = `https://${this.mediaDistribution.distributionDomainName}`;

// ============================================================================
// CLOUDFRONT - ASSETS DISTRIBUTION
// ============================================================================

const assetsOai = new cloudfront.OriginAccessIdentity(this, 'AssetsOai', {
  comment: `OAI for ${props.appName} assets bucket`,
});

this.assetsBucket.grantRead(assetsOai);

// Cache policy for static assets (long cache)
const assetsCachePolicy = new cloudfront.CachePolicy(this, 'AssetsCachePolicy', {
  cachePolicyName: `${resourcePrefix}-assets-cache`,
  defaultTtl: cdk.Duration.days(30),
  minTtl: cdk.Duration.days(1),
  maxTtl: cdk.Duration.days(365),
  enableAcceptEncodingGzip: true,
  enableAcceptEncodingBrotli: true,
  queryStringBehavior: cloudfront.CacheQueryStringBehavior.allowList('v'),
});

this.assetsDistribution = new cloudfront.Distribution(this, 'AssetsDistribution', {
  comment: `${props.appName} Assets Distribution`,

  defaultBehavior: {
    origin: new origins.S3Origin(this.assetsBucket, {
      originAccessIdentity: assetsOai,
    }),
    viewerProtocolPolicy: cloudfront.ViewerProtocolPolicy.REDIRECT_TO_HTTPS,
    allowedMethods: cloudfront.AllowedMethods.ALLOW_GET_HEAD,
    cachedMethods: cloudfront.CachedMethods.CACHE_GET_HEAD,
    cachePolicy: assetsCachePolicy,
    responseHeadersPolicy,
    compress: true,
  },

  // SPA support
  defaultRootObject: 'index.html',
```

```
    errorResponses: [
      {
        httpStatus: 404,
        responseHttpStatus: 200,
        responsePagePath: '/index.html',
        ttl: cdk.Duration.seconds(0),
      },
      {
        httpStatus: 403,
        responseHttpStatus: 200,
        responsePagePath: '/index.html',
        ttl: cdk.Duration.seconds(0),
      },
    ],

    priceClass: tierConfig.tier >= 4
      ? cloudfront.PriceClass.PRICE_CLASS_ALL
      : cloudfront.PriceClass.PRICE_CLASS_100,

    httpVersion: cloudfront.HttpVersion.HTTP2_AND_3,

    enableLogging: true,
    logBucket: this.logsBucket,
    logFilePrefix: 'cloudfront/assets/',

    minimumProtocolVersion: cloudfront.SecurityPolicyProtocol.TLS_V1_2_2021,
  });

  this.assetsUrl = `https://${this.assetsDistribution.distributionDomainName}`;

  // ============================================================================
  // S3 CROSS-REGION REPLICATION (Tier 4+)
  // ============================================================================

  if (props.enableReplication && tierConfig.s3.replicationEnabled) {
    // Note: Cross-region replication requires destination buckets in other regions
    // This would be handled by the multi-region deployment setup

    // Create replication role
    const replicationRole = new iam.Role(this, 'ReplicationRole', {
      assumedBy: new iam.ServicePrincipal('s3.amazonaws.com'),
      description: 'Role for S3 cross-region replication',
    });

    replicationRole.addToPolicy(new iam.PolicyStatement({
      actions: [
```

```
        's3:GetReplicationConfiguration',
        's3:ListBucket',
      ],
      resources: [this.mediaBucket.bucketArn],
    }));

    replicationRole.addToPolicy(new iam.PolicyStatement({
      actions: [
        's3:GetObjectVersionForReplication',
        's3:GetObjectVersionAcl',
        's3:GetObjectVersionTagging',
      ],
      resources: [`${this.mediaBucket.bucketArn}/*`],
    }));

    // Note: Destination bucket permissions would be added when those buckets are created
  }

  // ============================================================================
  // TAGS
  // ============================================================================

  applyTags(this, {
    appId: props.appId,
    environment: props.environment,
    tier: props.tier,
  });

  // ============================================================================
  // OUTPUTS
  // ============================================================================

  new cdk.CfnOutput(this, 'MediaBucketName', {
    value: this.mediaBucket.bucketName,
    description: 'Media S3 bucket name',
    exportName: `${resourcePrefix}-media-bucket`,
  });

  new cdk.CfnOutput(this, 'MediaBucketArn', {
    value: this.mediaBucket.bucketArn,
    description: 'Media S3 bucket ARN',
    exportName: `${resourcePrefix}-media-bucket-arn`,
  });

  new cdk.CfnOutput(this, 'AssetsBucketName', {
    value: this.assetsBucket.bucketName,
```

```
      description: 'Assets S3 bucket name',
      exportName: `${resourcePrefix}-assets-bucket`,
    });

    new cdk.CfnOutput(this, 'MediaDistributionId', {
      value: this.mediaDistribution.distributionId,
      description: 'Media CloudFront distribution ID',
      exportName: `${resourcePrefix}-media-cf-id`,
    });

    new cdk.CfnOutput(this, 'MediaDistributionUrl', {
      value: this.mediaUrl,
      description: 'Media CloudFront URL',
      exportName: `${resourcePrefix}-media-url`,
    });

    new cdk.CfnOutput(this, 'AssetsDistributionId', {
      value: this.assetsDistribution.distributionId,
      description: 'Assets CloudFront distribution ID',
      exportName: `${resourcePrefix}-assets-cf-id`,
    });

    new cdk.CfnOutput(this, 'AssetsDistributionUrl', {
      value: this.assetsUrl,
      description: 'Assets CloudFront URL',
      exportName: `${resourcePrefix}-assets-url`,
    });
  }
}
```

PART 9: STACK EXPORTS

packages/infrastructure/lib/stacks/index.ts

```
export * from './foundation.stack';
export * from './networking.stack';
export * from './security.stack';
export * from './data.stack';
export * from './storage.stack';
// Prompt 3 exports
// export * from './ai.stack';
// export * from './api.stack';
// export * from './admin.stack';
```

packages/infrastructure/lib/index.ts

```typescript
export * from './config';
export * from './stacks';
// export * from './constructs';  // When constructs are added
```

DEPLOYMENT COMMANDS

Development Environment (Tier 1)

```bash
cd packages/infrastructure

# Deploy all stacks
npx cdk deploy --all \
  --context appId=thinktank \
  --context appName="Think Tank" \
  --context environment=dev \
  --context tier=1 \
  --context domain=thinktank.YOUR_DOMAIN.com

# Deploy specific stack
npx cdk deploy thinktank-dev-networking \
  --context appId=thinktank \
  --context environment=dev \
  --context tier=1
```

Staging Environment (Tier 2)

```bash
npx cdk deploy --all \
  --context appId=thinktank \
  --context appName="Think Tank" \
  --context environment=staging \
  --context tier=2 \
  --context domain=thinktank.YOUR_DOMAIN.com
```

Production Environment (Tier 3-5)

```bash
# Tier 3 - Single Region
npx cdk deploy --all \
  --context appId=thinktank \
  --context appName="Think Tank" \
  --context environment=prod \
  --context tier=3 \
  --context domain=thinktank.YOUR_DOMAIN.com

# Tier 4 - Multi-Region
```

```bash
npx cdk deploy --all \
  --context appId=thinktank \
  --context appName="Think Tank" \
  --context environment=prod \
  --context tier=4 \
  --context domain=thinktank.YOUR_DOMAIN.com
```

STACK DEPENDENCIES

```
                    Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
                    Ã¢â€ â€š   Foundation    Ã¢â€ â€š
                    Ã¢â€ â€š     Stack       Ã¢â€ â€š
                    Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
                             Ã¢â€ â€š
                             Ã¢â€"Â¼
                    Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
                    Ã¢â€ â€š   Networking    Ã¢â€ â€š
                    Ã¢â€ â€š     Stack       Ã¢â€ â€š
                    Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
                             Ã¢â€ â€š
                             Ã¢â€"Â¼
                    Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
                    Ã¢â€ â€š    Security     Ã¢â€ â€š
                    Ã¢â€ â€š     Stack       Ã¢â€ â€š
                    Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
                             Ã¢â€ â€š
              Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
              Ã¢â€ â€š                    Ã¢â€ â€š                    Ã¢â€ â€š
              Ã¢â€"Â¼                    Ã¢â€"Â¼                    Ã¢â€"Â¼
     Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
     Ã¢â€ â€š    Data     Ã¢â€ â€š Ã¢â€ â€š    Storage    Ã¢â€ â€š Ã¢â€ â€š     AI       Ã¢â€ â€š
     Ã¢â€ â€š    Stack    Ã¢â€ â€š Ã¢â€ â€š     Stack     Ã¢â€ â€š Ã¢â€ â€š    Stack     Ã¢â€ â€š
     Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ Â¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
            Ã¢â€ â€š                                    Ã¢â€ â€š
            Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€
                            Ã¢â€ â€š
                            Ã¢â€"Â¼
                   Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
                   Ã¢â€ â€š      API         Ã¢â€ â€š
                   Ã¢â€ â€š     Stack        Ã¢â€ â€š
                   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã
                            Ã¢â€ â€š
                            Ã¢â€"Â¼
                   Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
```

```
Ã¢â€ â€š     Admin       Ã¢â€ â€š
Ã¢â€ â€š     Stack       Ã¢â€ â€š
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â
```

NEXT PROMPTS

Continue with: - **Prompt 3**: CDK AI & API Stacks (LiteLLM, SageMaker, API Gateway, Cognito) - **Prompt 4**: Lambda Functions - Core (Router, Providers, Models, Chat) - **Prompt 5**: Lambda Functions - Admin & Billing (Invitations, Approvals, Metering) - **Prompt 6**: Self-Hosted Models & Mid-Level Services - **Prompt 7**: External Providers & Database Schema/Migrations - **Prompt 8**: Admin Web Dashboard (Next.js) - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 2: CDK Infrastructure Stacks RADIANT v2.2.0 - December 2024

â•

END OF SECTION 2

â•

â•

SECTION-03-CDK-AI-AND-API-STACKS

SECTION 3: CDK AI & API STACKS (v2.1.0)

â•

> **Dependencies:** Sections 0-2 **Creates:** Cognito, LiteLLM, API Gateway, AppSync, Admin stacks

Type Imports

```
import {
  TierConfig,
  getTierConfig,
  EXTERNAL_PROVIDERS,
```

```
  PROVIDER_ENDPOINTS,
} from '@radiant/shared';
```

RADIANT v2.2.0 - Prompt 3: CDK AI & API Stacks

> **Prompt 3 of 9** | Target Size: ~50KB | Version: 3.7.0 | December 2024

OVERVIEW

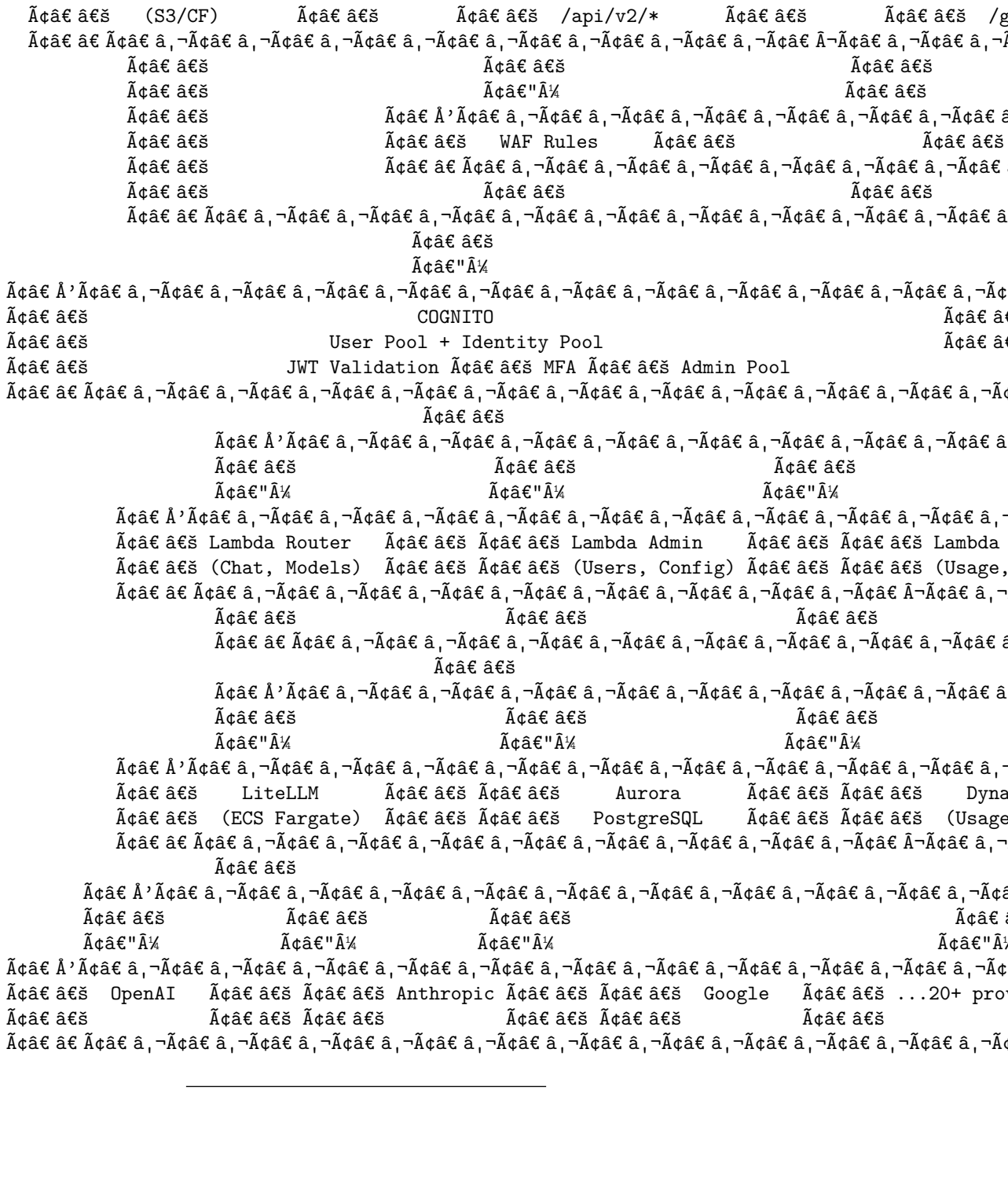
This prompt creates the AI services and API layer stacks:

1. **Auth Stack** - Cognito User Pool, Identity Pool, MFA configuration
2. **AI Stack** - LiteLLM on ECS Fargate, SageMaker endpoints for self-hosted models
3. **API Stack** - API Gateway REST API, AppSync GraphQL, Lambda integrations
4. **Admin Stack** - Admin dashboard hosting, admin-specific APIs

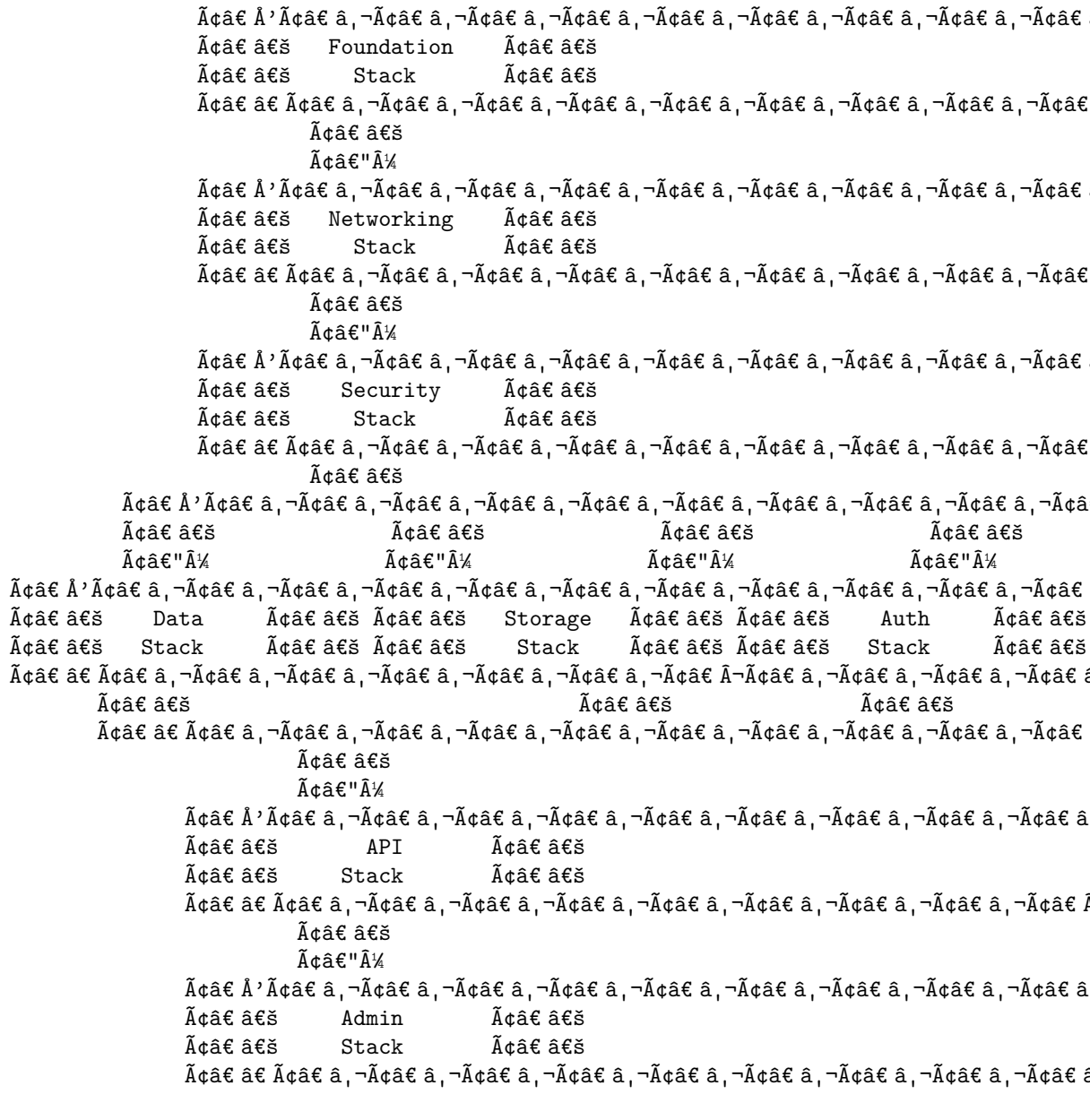
All stacks are tier-aware with automatic scaling based on infrastructure tier (1-5).

ARCHITECTURE OVERVIEW

```
Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
Ã¢â€ â€š                              CLIENT LAYER                              Ã¢â€ â
Ã¢â€ â€š     Web Apps Ã¢â€ â€š Mobile Apps Ã¢â€ â€š Desktop Apps Ã¢â€ â€š Admin Dashboard
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
                                     Ã¢â€ â€š HTTPS
                                     Ã¢â€"Â¼
Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
Ã¢â€ â€š                           CLOUDFRONT (CDN)                           Ã¢â€ â
Ã¢â€ â€š         TLS 1.3 Ã¢â€ â€š Geographic Routing Ã¢â€ â€š Caching Ã¢â€ â€š DDoS Protectio
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
                                     Ã¢â€ â€š
           Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
           Ã¢â€ â€š                                Ã¢â€ â€š                              Ã¢â€ â€š
           Ã¢â€"Â¼                                Ã¢â€"Â¼                              Ã¢â€"Â¼
  Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
  Ã¢â€ â€š   Admin SPA     Ã¢â€ â€š       Ã¢â€ â€š  REST API GW    Ã¢â€ â€š       Ã¢â€ â€š  Ap
```


(S3/CF)
/api/v2/*
WAF Rules
COGNITO
User Pool + Identity Pool
JWT Validation
MFA
Admin Pool
Lambda Router
(Chat, Models)
Lambda Admin
(Users, Config)
Lambda
(Usage,
LiteLLM
(ECS Fargate)
Aurora
PostgreSQL
Dyna
(Usage
OpenAI
Anthropic
Google
...20+ pro

STACK DEPENDENCIES



PART 1: UPDATE CDK ENTRY POINT

Update `packages/infrastructure/bin/radiant.ts` to include all stacks:

packages/infrastructure/bin/radiant.ts (Complete)

```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import { getTierConfig, TierLevel } from '../lib/config/tiers';
import { MULTI_REGION_CONFIG } from '../lib/config/regions';

// Import all stacks
import { FoundationStack } from '../lib/stacks/foundation.stack';
import { NetworkingStack } from '../lib/stacks/networking.stack';
import { SecurityStack } from '../lib/stacks/security.stack';
import { DataStack } from '../lib/stacks/data.stack';
import { StorageStack } from '../lib/stacks/storage.stack';
import { AuthStack } from '../lib/stacks/auth.stack';
import { AIStack } from '../lib/stacks/ai.stack';
import { APIStack } from '../lib/stacks/api.stack';
import { AdminStack } from '../lib/stacks/admin.stack';

const app = new cdk.App();

// ============================================================================
// CONFIGURATION
// ============================================================================

const appId = app.node.tryGetContext('appId') || 'radiant';
const appName = app.node.tryGetContext('appName') || 'Radiant';
const environment = app.node.tryGetContext('environment') || 'dev';
const tier = (app.node.tryGetContext('tier') || 1) as TierLevel;
const domain = app.node.tryGetContext('domain') || `${appId}.YOUR_DOMAIN.com`;
const hostedZoneId = app.node.tryGetContext('hostedZoneId');

const tierConfig = getTierConfig(tier);

// Determine regions based on tier
const primaryRegion = process.env.CDK_DEFAULT_REGION || 'us-east-1';
const enableMultiRegion = tierConfig.features.multiRegion && environment === 'prod';
const regions = enableMultiRegion
  ? [primaryRegion, ...Object.values(MULTI_REGION_CONFIG).filter(r => r !== primaryRegion)]
  : [primaryRegion];

// ============================================================================
// LOGGING
// ============================================================================

console.log('Ã¢â€¢â€ Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â
```

```
console.log('Ã¢â€¢â€˜                    RADIANT DEPLOYMENT                          Ã¢â€¢â€
console.log('Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â
console.log(`Ã¢â€¢â€˜ App ID:       ${appId.padEnd(48)}Ã¢â€¢â€˜`);
console.log(`Ã¢â€¢â€˜ Environment:  ${environment.padEnd(48)}Ã¢â€¢â€˜`);
console.log(`Ã¢â€¢â€˜ Tier:         ${tier} - ${tierConfig.name.padEnd(42)}Ã¢â€¢â€˜`);
console.log(`Ã¢â€¢â€˜ Domain:       ${domain.padEnd(48)}Ã¢â€¢â€˜`);
console.log(`Ã¢â€¢â€˜ Regions:      ${regions.join(', ').padEnd(48)}Ã¢â€¢â€˜`);
console.log(`Ã¢â€¢â€˜ Multi-Region: ${(enableMultiRegion ? 'Yes' : 'No').padEnd(48)}Ã¢â€¢â€
console.log('Ã¢â€¢Å¡Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â Ã¢â€¢Â

// ============================================================================
// STACK PREFIX
// ============================================================================

const stackPrefix = `${appId}-${environment}`;

const env = {
  account: process.env.CDK_DEFAULT_ACCOUNT,
  region: primaryRegion
};

const commonProps = {
  appId,
  appName,
  environment,
  tier,
  tierConfig,
  domain,
};

// ============================================================================
// FOUNDATION STACKS (From Prompt 2)
// ============================================================================

// 1. Foundation Stack
const foundationStack = new FoundationStack(app, `${stackPrefix}-foundation`, {
  env,
  ...commonProps,
  description: `RADIANT Foundation - ${appName} ${environment}`,
});

// 2. Networking Stack
const networkingStack = new NetworkingStack(app, `${stackPrefix}-networking`, {
  env,
  ...commonProps,
  description: `RADIANT Networking - ${appName} ${environment}`,
```

```
});
networkingStack.addDependency(foundationStack);

// 3. Security Stack
const securityStack = new SecurityStack(app, `${stackPrefix}-security`, {
  env,
  ...commonProps,
  vpc: networkingStack.vpc,
  description: `RADIANT Security - ${appName} ${environment}`,
});
securityStack.addDependency(networkingStack);

// 4. Data Stack
const dataStack = new DataStack(app, `${stackPrefix}-data`, {
  env,
  ...commonProps,
  vpc: networkingStack.vpc,
  databaseSecurityGroup: securityStack.databaseSecurityGroup,
  lambdaSecurityGroup: securityStack.lambdaSecurityGroup,
  dataKey: securityStack.dataKey,
  isPrimary: true,
  enableGlobal: enableMultiRegion,
  description: `RADIANT Data Layer - ${appName} ${environment}`,
});
dataStack.addDependency(securityStack);

// 5. Storage Stack
const storageStack = new StorageStack(app, `${stackPrefix}-storage`, {
  env,
  ...commonProps,
  vpc: networkingStack.vpc,
  mediaKey: securityStack.mediaKey,
  hostedZoneId,
  enableReplication: enableMultiRegion,
  description: `RADIANT Storage - ${appName} ${environment}`,
});
storageStack.addDependency(securityStack);

// ============================================================================
// AI & API STACKS (Prompt 3)
// ============================================================================

// 6. Auth Stack (Cognito)
const authStack = new AuthStack(app, `${stackPrefix}-auth`, {
  env,
  ...commonProps,
```

```
  description: `RADIANT Auth - ${appName} ${environment}`,
});
authStack.addDependency(securityStack);

// 7. AI Stack (LiteLLM + SageMaker)
const aiStack = new AIStack(app, `${stackPrefix}-ai`, {
  env,
  ...commonProps,
  vpc: networkingStack.vpc,
  ecsSecurityGroup: securityStack.ecsSecurityGroup,
  lambdaSecurityGroup: securityStack.lambdaSecurityGroup,
  albSecurityGroup: securityStack.albSecurityGroup,
  secretsKey: securityStack.secretsKey,
  description: `RADIANT AI Services - ${appName} ${environment}`,
});
aiStack.addDependency(securityStack);
aiStack.addDependency(networkingStack);

// 8. API Stack (REST + GraphQL)
const apiStack = new APIStack(app, `${stackPrefix}-api`, {
  env,
  ...commonProps,
  vpc: networkingStack.vpc,
  lambdaSecurityGroup: securityStack.lambdaSecurityGroup,
  userPool: authStack.userPool,
  userPoolClient: authStack.userPoolClient,
  litellmUrl: aiStack.litellmUrl,
  auroraCluster: dataStack.cluster,
  auroraSecret: dataStack.dbSecret,
  usageTable: dataStack.usageTable,
  sessionsTable: dataStack.sessionsTable,
  cacheTable: dataStack.cacheTable,
  mediaBucket: storageStack.mediaBucket,
  lambdaRole: securityStack.lambdaExecutionRole,
  webAcl: securityStack.webAcl,
  description: `RADIANT API Layer - ${appName} ${environment}`,
});
apiStack.addDependency(authStack);
apiStack.addDependency(aiStack);
apiStack.addDependency(dataStack);
apiStack.addDependency(storageStack);

// 9. Admin Stack
const adminStack = new AdminStack(app, `${stackPrefix}-admin`, {
  env,
  ...commonProps,
```

```
  vpc: networkingStack.vpc,
  adminUserPool: authStack.adminUserPool,
  adminUserPoolClient: authStack.adminUserPoolClient,
  restApi: apiStack.api,
  graphqlApi: apiStack.graphqlApi,
  assetsBucket: storageStack.assetsBucket,
  assetsDistribution: storageStack.assetsDistribution,
  hostedZoneId,
  description: `RADIANT Admin Dashboard - ${appName} ${environment}`,
});
adminStack.addDependency(apiStack);

// ============================================================================
// TAGGING
// ============================================================================

cdk.Tags.of(app).add('Project', 'RADIANT');
cdk.Tags.of(app).add('AppId', appId);
cdk.Tags.of(app).add('Environment', environment);
cdk.Tags.of(app).add('Tier', String(tier));
cdk.Tags.of(app).add('ManagedBy', 'CDK');
cdk.Tags.of(app).add('Version', '2.2.0');
```

PART 2: AUTH STACK (COGNITO)

packages/infrastructure/lib/stacks/auth.stack.ts

```
import * as cdk from 'aws-cdk-lib';
import * as cognito from 'aws-cdk-lib/aws-cognito';
import * as iam from 'aws-cdk-lib/aws-iam';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface AuthStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
}

/**
```

```
 * Auth Stack
 *
 * Creates Cognito User Pools for both end-users and administrators.
 * Configures MFA, password policies, and Identity Pool for AWS access.
 */
export class AuthStack extends cdk.Stack {
  // User Pool (End Users)
  public readonly userPool: cognito.UserPool;
  public readonly userPoolClient: cognito.UserPoolClient;
  public readonly userPoolDomain: cognito.UserPoolDomain;

  // Admin User Pool (Administrators)
  public readonly adminUserPool: cognito.UserPool;
  public readonly adminUserPoolClient: cognito.UserPoolClient;
  public readonly adminUserPoolDomain: cognito.UserPoolDomain;

  // Identity Pool
  public readonly identityPool: cognito.CfnIdentityPool;

  // Domain
  public readonly cognitoDomain: string;

  constructor(scope: Construct, id: string, props: AuthStackProps) {
    super(scope, id, props);

    const { tierConfig } = props;
    const resourcePrefix = `${props.appId}-${props.environment}`;

    // ========================================================================
    // END-USER COGNITO USER POOL
    // ========================================================================

    this.userPool = new cognito.UserPool(this, 'UserPool', {
      userPoolName: `${resourcePrefix}-users`,

      // Sign-in configuration
      signInAliases: {
        email: true,
        username: false,
      },

      // Self-registration
      selfSignUpEnabled: true,

      // Verification
      autoVerify: {
```

```
    email: true,
  },

  // User verification
  userVerification: {
    emailSubject: `Welcome to ${props.appName} - Verify your email`,
    emailBody: `
      <h2>Welcome to ${props.appName}!</h2>
      <p>Thank you for signing up. Please verify your email address by entering this coo
      <h1 style="color: #4F46E5; letter-spacing: 4px;">{####}</h1>
      <p>This code expires in 24 hours.</p>
      <p>If you didn't create this account, please ignore this email.</p>
    `,
    emailStyle: cognito.VerificationEmailStyle.CODE,
  },

  // Password policy
  passwordPolicy: {
    minLength: 12,
    requireLowercase: true,
    requireUppercase: true,
    requireDigits: true,
    requireSymbols: true,
    tempPasswordValidity: cdk.Duration.days(7),
  },

  // MFA configuration
  mfa: tierConfig.tier >= 3
    ? cognito.Mfa.OPTIONAL
    : cognito.Mfa.OFF,
  mfaSecondFactor: {
    sms: true,
    otp: true,
  },

  // Account recovery
  accountRecovery: cognito.AccountRecovery.EMAIL_ONLY,

  // Standard attributes
  standardAttributes: {
    email: {
      required: true,
      mutable: true,
    },
    fullname: {
      required: false,
```

```
      mutable: true,
    },
  },

  // Custom attributes
  customAttributes: {
    tenantId: new cognito.StringAttribute({ mutable: false }),
    role: new cognito.StringAttribute({ mutable: true }),
    createdAt: new cognito.StringAttribute({ mutable: false }),
  },

  // Device tracking
  deviceTracking: {
    challengeRequiredOnNewDevice: tierConfig.tier >= 3,
    deviceOnlyRememberedOnUserPrompt: true,
  },

  // Advanced security (Tier 3+)
  advancedSecurityMode: tierConfig.tier >= 3
    ? cognito.AdvancedSecurityMode.ENFORCED
    : cognito.AdvancedSecurityMode.OFF,

  // Removal policy
  removalPolicy: props.environment === 'prod'
    ? cdk.RemovalPolicy.RETAIN
    : cdk.RemovalPolicy.DESTROY,
});

// User Pool Client
this.userPoolClient = this.userPool.addClient('UserPoolClient', {
  userPoolClientName: `${resourcePrefix}-web-client`,

  // OAuth configuration
  oAuth: {
    flows: {
      authorizationCodeGrant: true,
      implicitCodeGrant: false,
    },
    scopes: [
      cognito.OAuthScope.EMAIL,
      cognito.OAuthScope.OPENID,
      cognito.OAuthScope.PROFILE,
    ],
    callbackUrls: [
      `https://${props.domain}/auth/callback`,
      `https://admin.${props.domain}/auth/callback`,
```

```
        'http://localhost:3000/auth/callback',
      ],
      logoutUrls: [
        `https://${props.domain}/auth/logout`,
        `https://admin.${props.domain}/auth/logout`,
        'http://localhost:3000/auth/logout',
      ],
    },

    // Token configuration
    accessTokenValidity: cdk.Duration.hours(1),
    idTokenValidity: cdk.Duration.hours(1),
    refreshTokenValidity: cdk.Duration.days(30),

    // Auth flows
    authFlows: {
      userPassword: true,
      userSrp: true,
      custom: false,
    },

    // Prevent user existence errors
    preventUserExistenceErrors: true,

    // Generate client secret for server-side apps
    generateSecret: false,
  });

  // User Pool Domain
  const domainPrefix = `${props.appId}-${props.environment}-${this.account}`.toLowerCase()
  this.userPoolDomain = this.userPool.addDomain('UserPoolDomain', {
    cognitoDomain: {
      domainPrefix,
    },
  });
  this.cognitoDomain = `${domainPrefix}.auth.${this.region}.amazoncognito.com`;

  // ============================================================================
  // ADMIN COGNITO USER POOL (Separate for enhanced security)
  // ============================================================================

  this.adminUserPool = new cognito.UserPool(this, 'AdminUserPool', {
    userPoolName: `${resourcePrefix}-admins`,

    // Sign-in
    signInAliases: {
```

```
  email: true,
  username: false,
},

// No self-registration for admins
selfSignUpEnabled: false,

// Verification
autoVerify: {
  email: true,
},

// Invitation settings
userInvitation: {
  emailSubject: `You're invited to ${props.appName} Admin`,
  emailBody: `
    <h2>Welcome to ${props.appName} Admin!</h2>
    <p>You have been invited to join as an administrator.</p>
    <p>Your temporary password is: <strong>{####}</strong></p>
    <p>Please sign in at <a href="https://admin.${props.domain}">admin.${props.domain}
    <p>This invitation expires in 7 days.</p>
  `,
},

// Strict password policy for admins
passwordPolicy: {
  minLength: 16,
  requireLowercase: true,
  requireUppercase: true,
  requireDigits: true,
  requireSymbols: true,
  tempPasswordValidity: cdk.Duration.days(7),
},

// MFA REQUIRED for production admins
mfa: props.environment === 'prod'
  ? cognito.Mfa.REQUIRED
  : cognito.Mfa.OPTIONAL,
mfaSecondFactor: {
  sms: false,  // TOTP only for admins
  otp: true,
},

// Account recovery
accountRecovery: cognito.AccountRecovery.EMAIL_ONLY,
```

```
  // Standard attributes
  standardAttributes: {
    email: {
      required: true,
      mutable: true,
    },
    fullname: {
      required: true,
      mutable: true,
    },
  },

  // Custom attributes
  customAttributes: {
    adminRole: new cognito.StringAttribute({ mutable: true }),
    invitedBy: new cognito.StringAttribute({ mutable: false }),
    invitedAt: new cognito.StringAttribute({ mutable: false }),
  },

  // Always track admin devices
  deviceTracking: {
    challengeRequiredOnNewDevice: true,
    deviceOnlyRememberedOnUserPrompt: true,
  },

  // Always enforce advanced security for admins
  advancedSecurityMode: cognito.AdvancedSecurityMode.ENFORCED,

  // Never delete admin pool
  removalPolicy: cdk.RemovalPolicy.RETAIN,
});

// Admin User Pool Groups
new cognito.CfnUserPoolGroup(this, 'SuperAdminsGroup', {
  userPoolId: this.adminUserPool.userPoolId,
  groupName: 'super_admin',
  description: 'Full platform access including user management',
  precedence: 1,
});

new cognito.CfnUserPoolGroup(this, 'AdminsGroup', {
  userPoolId: this.adminUserPool.userPoolId,
  groupName: 'admin',
  description: 'Can deploy and manage assigned applications',
  precedence: 2,
});
```

```
new cognito.CfnUserPoolGroup(this, 'OperatorsGroup', {
  userPoolId: this.adminUserPool.userPoolId,
  groupName: 'operator',
  description: 'Read-only access with audit log viewing',
  precedence: 3,
});

new cognito.CfnUserPoolGroup(this, 'AuditorsGroup', {
  userPoolId: this.adminUserPool.userPoolId,
  groupName: 'auditor',
  description: 'Billing and audit log access only',
  precedence: 4,
});

// Admin User Pool Client
this.adminUserPoolClient = this.adminUserPool.addClient('AdminUserPoolClient', {
  userPoolClientName: `${resourcePrefix}-admin-client`,

  oAuth: {
    flows: {
      authorizationCodeGrant: true,
      implicitCodeGrant: false,
    },
    scopes: [
      cognito.OAuthScope.EMAIL,
      cognito.OAuthScope.OPENID,
      cognito.OAuthScope.PROFILE,
    ],
    callbackUrls: [
      `https://admin.${props.domain}/auth/callback`,
      'http://localhost:3001/auth/callback',
    ],
    logoutUrls: [
      `https://admin.${props.domain}/auth/logout`,
      'http://localhost:3001/auth/logout',
    ],
  },

  // Shorter token validity for admins
  accessTokenValidity: cdk.Duration.minutes(30),
  idTokenValidity: cdk.Duration.minutes(30),
  refreshTokenValidity: cdk.Duration.days(7),

  authFlows: {
    userPassword: false,  // SRP only for admins
```

```
    userSrp: true,
    custom: false,
  },

  preventUserExistenceErrors: true,
  generateSecret: false,
});

// Admin User Pool Domain
const adminDomainPrefix = `${props.appId}-${props.environment}-admin-${this.account}`.to
this.adminUserPoolDomain = this.adminUserPool.addDomain('AdminUserPoolDomain', {
  cognitoDomain: {
    domainPrefix: adminDomainPrefix,
  },
});

// ============================================================================
// IDENTITY POOL (Federated Identities)
// ============================================================================

this.identityPool = new cognito.CfnIdentityPool(this, 'IdentityPool', {
  identityPoolName: `${resourcePrefix.replace(/-/g, '_')}_identity`,

  allowUnauthenticatedIdentities: false,

  cognitoIdentityProviders: [
    {
      clientId: this.userPoolClient.userPoolClientId,
      providerName: this.userPool.userPoolProviderName,
      serverSideTokenCheck: true,
    },
  ],
});

// Authenticated Role
const authenticatedRole = new iam.Role(this, 'AuthenticatedRole', {
  assumedBy: new iam.FederatedPrincipal(
    'cognito-identity.amazonaws.com',
    {
      StringEquals: {
        'cognito-identity.amazonaws.com:aud': this.identityPool.ref,
      },
      'ForAnyValue:StringLike': {
        'cognito-identity.amazonaws.com:amr': 'authenticated',
      },
    },
```

```
      'sts:AssumeRoleWithWebIdentity'
    ),
    description: 'Role for authenticated users',
  });

  // Basic permissions for authenticated users
  authenticatedRole.addToPolicy(new iam.PolicyStatement({
    actions: [
      's3:GetObject',
      's3:PutObject',
    ],
    resources: [
      `arn:aws:s3:::${props.appId}-${props.environment}-media-*/*`,
    ],
    conditions: {
      StringLike: {
        's3:prefix': ['uploads/${cognito-identity.amazonaws.com:sub}/*'],
      },
    },
  }));

  // Attach roles to Identity Pool
  new cognito.CfnIdentityPoolRoleAttachment(this, 'IdentityPoolRoles', {
    identityPoolId: this.identityPool.ref,
    roles: {
      authenticated: authenticatedRole.roleArn,
    },
  });

  // ============================================================================
  // SSM PARAMETERS
  // ============================================================================

  new ssm.StringParameter(this, 'UserPoolIdParam', {
    parameterName: `/radiant/${props.appId}/${props.environment}/auth/user-pool-id`,
    stringValue: this.userPool.userPoolId,
  });

  new ssm.StringParameter(this, 'UserPoolClientIdParam', {
    parameterName: `/radiant/${props.appId}/${props.environment}/auth/user-pool-client-id`
    stringValue: this.userPoolClient.userPoolClientId,
  });

  new ssm.StringParameter(this, 'AdminUserPoolIdParam', {
    parameterName: `/radiant/${props.appId}/${props.environment}/auth/admin-user-pool-id`,
    stringValue: this.adminUserPool.userPoolId,
```

```
});

new ssm.StringParameter(this, 'AdminUserPoolClientIdParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/auth/admin-user-pool-clie
  stringValue: this.adminUserPoolClient.userPoolClientId,
});

new ssm.StringParameter(this, 'CognitoDomainParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/auth/cognito-domain`,
  stringValue: this.cognitoDomain,
});

new ssm.StringParameter(this, 'IdentityPoolIdParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/auth/identity-pool-id`,
  stringValue: this.identityPool.ref,
});

// ============================================================================
// TAGS
// ============================================================================

applyTags(this, {
  appId: props.appId,
  environment: props.environment,
  tier: props.tier,
});

// ============================================================================
// OUTPUTS
// ============================================================================

new cdk.CfnOutput(this, 'UserPoolId', {
  value: this.userPool.userPoolId,
  description: 'Cognito User Pool ID',
  exportName: `${resourcePrefix}-user-pool-id`,
});

new cdk.CfnOutput(this, 'UserPoolClientId', {
  value: this.userPoolClient.userPoolClientId,
  description: 'Cognito User Pool Client ID',
  exportName: `${resourcePrefix}-user-pool-client-id`,
});

new cdk.CfnOutput(this, 'CognitoDomainOutput', {
  value: this.cognitoDomain,
  description: 'Cognito Domain',
```

```
      exportName: `${resourcePrefix}-cognito-domain`,
    });

    new cdk.CfnOutput(this, 'IdentityPoolId', {
      value: this.identityPool.ref,
      description: 'Cognito Identity Pool ID',
      exportName: `${resourcePrefix}-identity-pool-id`,
    });

    new cdk.CfnOutput(this, 'AdminUserPoolId', {
      value: this.adminUserPool.userPoolId,
      description: 'Admin Cognito User Pool ID',
      exportName: `${resourcePrefix}-admin-user-pool-id`,
    });

    new cdk.CfnOutput(this, 'AdminUserPoolClientId', {
      value: this.adminUserPoolClient.userPoolClientId,
      description: 'Admin Cognito User Pool Client ID',
      exportName: `${resourcePrefix}-admin-user-pool-client-id`,
    });
  }
}
```

PART 3: AI STACK (LITELLM + SAGEMAKER)

packages/infrastructure/lib/stacks/ai.stack.ts

```
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecsPatterns from 'aws-cdk-lib/aws-ecs-patterns';
import * as logs from 'aws-cdk-lib/aws-logs';
import * as secretsmanager from 'aws-cdk-lib/aws-secretsmanager';
import * as iam from 'aws-cdk-lib/aws-iam';
import * as kms from 'aws-cdk-lib/aws-kms';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import * as sagemaker from 'aws-cdk-lib/aws-sagemaker';
import * as applicationautoscaling from 'aws-cdk-lib/aws-applicationautoscaling';
import * as servicediscovery from 'aws-cdk-lib/aws-servicediscovery';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface AIStackProps extends cdk.StackProps {
  appId: string;
```

```
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
  vpc: ec2.Vpc;
  ecsSecurityGroup: ec2.SecurityGroup;
  lambdaSecurityGroup: ec2.SecurityGroup;
  albSecurityGroup: ec2.SecurityGroup;
  secretsKey: kms.Key;
}

/**
 * AI Stack
 *
 * Creates LiteLLM proxy on ECS Fargate for unified AI routing.
 * Deploys SageMaker endpoints for self-hosted models (Tier 3+).
 * Supports 20+ external providers and 30+ self-hosted models.
 */
export class AIStack extends cdk.Stack {
  // ECS Resources
  public readonly cluster: ecs.Cluster;
  public readonly litellmService: ecsPatterns.ApplicationLoadBalancedFargateService;
  public readonly litellmUrl: string;

  // Service Discovery
  public readonly namespace: servicediscovery.PrivateDnsNamespace;

  // Secrets
  public readonly providerSecretsArn: string;

  // SageMaker (Tier 3+)
  public readonly sagemakerRole?: iam.Role;
  public readonly sagemakerEndpoints: Map<string, sagemaker.CfnEndpoint> = new Map();

  constructor(scope: Construct, id: string, props: AIStackProps) {
    super(scope, id, props);

    const { tierConfig, vpc } = props;
    const resourcePrefix = `${props.appId}-${props.environment}`;

    // ============================================================================
    // SERVICE DISCOVERY NAMESPACE
    // ============================================================================

    this.namespace = new servicediscovery.PrivateDnsNamespace(this, 'AiNamespace', {
```

```
  name: `${resourcePrefix}.ai.internal`,
  vpc,
  description: `Service discovery namespace for ${props.appName} AI services`,
});

// ============================================================================
// PROVIDER API KEYS SECRET
// ============================================================================

const providerSecrets = new secretsmanager.Secret(this, 'ProviderSecrets', {
  secretName: `${resourcePrefix}/ai/provider-keys`,
  description: `API keys for AI providers - ${props.appName}`,
  encryptionKey: props.secretsKey,
  generateSecretString: {
    secretStringTemplate: JSON.stringify({
      OPENAI_API_KEY: '',
      ANTHROPIC_API_KEY: '',
      GOOGLE_API_KEY: '',
      XAI_API_KEY: '',
      DEEPSEEK_API_KEY: '',
      PERPLEXITY_API_KEY: '',
      COHERE_API_KEY: '',
      MISTRAL_API_KEY: '',
      GROQ_API_KEY: '',
      TOGETHER_API_KEY: '',
      FIREWORKS_API_KEY: '',
      STABILITY_API_KEY: '',
      ELEVENLABS_API_KEY: '',
      RUNWAY_API_KEY: '',
      REPLICATE_API_TOKEN: '',
      MESHY_API_KEY: '',
      AZURE_OPENAI_API_KEY: '',
      AZURE_OPENAI_ENDPOINT: '',
      AWS_BEDROCK_REGION: props.env?.region || 'us-east-1',
    }),
    generateStringKey: 'LITELLM_MASTER_KEY',
    excludeCharacters: '"\'\\',
  },
});
this.providerSecretsArn = providerSecrets.secretArn;

// ============================================================================
// ECS CLUSTER
// ============================================================================

this.cluster = new ecs.Cluster(this, 'AiCluster', {
```

```
  clusterName: `${resourcePrefix}-ai`,
  vpc,
  containerInsights: tierConfig.tier >= 2,
  enableFargateCapacityProviders: true,
});

// ============================================================================
// LITELLM TASK DEFINITION
// ============================================================================

const litellmLogGroup = new logs.LogGroup(this, 'LitellmLogs', {
  logGroupName: `/radiant/${props.appId}/${props.environment}/litellm`,
  retention: tierConfig.tier >= 3
    ? logs.RetentionDays.THREE_MONTHS
    : logs.RetentionDays.TWO_WEEKS,
  removalPolicy: cdk.RemovalPolicy.DESTROY,
});

const litellmTaskDef = new ecs.FargateTaskDefinition(this, 'LitellmTaskDef', {
  family: `${resourcePrefix}-litellm`,
  cpu: tierConfig.litellm.cpu,
  memoryLimitMiB: tierConfig.litellm.memory,
});

// Grant secrets access
providerSecrets.grantRead(litellmTaskDef.taskRole);

// LiteLLM Container
const litellmContainer = litellmTaskDef.addContainer('litellm', {
  image: ecs.ContainerImage.fromRegistry('ghcr.io/berriai/litellm:main-latest'),
  essential: true,
  logging: ecs.LogDrivers.awsLogs({
    streamPrefix: 'litellm',
    logGroup: litellmLogGroup,
  }),
  environment: {
    LITELLM_LOG: 'DEBUG',
    DATABASE_URL: '', // Will be set by Lambda or config
    STORE_MODEL_IN_DB: 'true',
    UI_USERNAME: 'admin',
  },
  secrets: {
    LITELLM_MASTER_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'LITELLM_MASTER_K
    OPENAI_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'OPENAI_API_KEY'),
    ANTHROPIC_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'ANTHROPIC_API_KEY
    GOOGLE_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'GOOGLE_API_KEY'),
```

```
    XAI_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'XAI_API_KEY'),
    DEEPSEEK_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'DEEPSEEK_API_KEY'
    PERPLEXITY_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'PERPLEXITY_API_K
    COHERE_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'COHERE_API_KEY'),
    MISTRAL_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'MISTRAL_API_KEY'),
    GROQ_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'GROQ_API_KEY'),
    TOGETHER_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'TOGETHER_API_KEY'
    FIREWORKS_API_KEY: ecs.Secret.fromSecretsManager(providerSecrets, 'FIREWORKS_API_KEY
  },
  healthCheck: {
    command: ['CMD-SHELL', 'curl -f http://localhost:4000/health || exit 1'],
    interval: cdk.Duration.seconds(30),
    timeout: cdk.Duration.seconds(5),
    retries: 3,
    startPeriod: cdk.Duration.seconds(60),
  },
});

litellmContainer.addPortMappings({
  containerPort: 4000,
  protocol: ecs.Protocol.TCP,
});

// ============================================================================
// LITELLM FARGATE SERVICE WITH ALB
// ============================================================================

this.litellmService = new ecsPatterns.ApplicationLoadBalancedFargateService(this, 'Litel
  cluster: this.cluster,
  serviceName: `${resourcePrefix}-litellm`,
  taskDefinition: litellmTaskDef,

  // Capacity
  desiredCount: tierConfig.litellm.taskCount,

  // Networking
  assignPublicIp: false,
  securityGroups: [props.ecsSecurityGroup],
  taskSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },

  // Load Balancer
  publicLoadBalancer: false,
  loadBalancerName: `${resourcePrefix}-litellm-alb`,

  // Service discovery
  cloudMapOptions: {
```

```
    name: 'litellm',
    cloudMapNamespace: this.namespace,
    dnsRecordType: servicediscovery.DnsRecordType.A,
  },

  // Health check
  healthCheckGracePeriod: cdk.Duration.seconds(120),

  // Capacity providers
  capacityProviderStrategies: [
    {
      capacityProvider: 'FARGATE',
      weight: 1,
      base: 1,
    },
    {
      capacityProvider: 'FARGATE_SPOT',
      weight: tierConfig.tier >= 3 ? 2 : 0,
    },
  ],
});

// Configure health check
this.litellmService.targetGroup.configureHealthCheck({
  path: '/health',
  healthyHttpCodes: '200',
  healthyThresholdCount: 2,
  unhealthyThresholdCount: 3,
  interval: cdk.Duration.seconds(30),
  timeout: cdk.Duration.seconds(10),
});

// Internal URL
this.litellmUrl = `http://${this.litellmService.loadBalancer.loadBalancerDnsName}`;

// ============================================================================
// AUTO SCALING (Tier 2+)
// ============================================================================

if (tierConfig.litellm.enableAutoScaling) {
  const scaling = this.litellmService.service.autoScaleTaskCount({
    minCapacity: tierConfig.litellm.minTasks || 1,
    maxCapacity: tierConfig.litellm.maxTasks || tierConfig.litellm.taskCount * 2,
  });

  // Scale on CPU
```

```
    scaling.scaleOnCpuUtilization('CpuScaling', {
      targetUtilizationPercent: 70,
      scaleInCooldown: cdk.Duration.minutes(5),
      scaleOutCooldown: cdk.Duration.minutes(2),
    });

    // Scale on Memory
    scaling.scaleOnMemoryUtilization('MemoryScaling', {
      targetUtilizationPercent: 80,
      scaleInCooldown: cdk.Duration.minutes(5),
      scaleOutCooldown: cdk.Duration.minutes(2),
    });

    // Scale on request count (Tier 3+)
    if (tierConfig.tier >= 3) {
      scaling.scaleOnRequestCount('RequestScaling', {
        requestsPerTarget: 1000,
        targetGroup: this.litellmService.targetGroup,
        scaleInCooldown: cdk.Duration.minutes(5),
        scaleOutCooldown: cdk.Duration.minutes(1),
      });
    }
  }

  // ============================================================================
  // SAGEMAKER ENDPOINTS (Tier 3+)
  // ============================================================================

  if (tierConfig.sagemaker.enabled) {
    this.sagemakerRole = new iam.Role(this, 'SageMakerRole', {
      roleName: `${resourcePrefix}-sagemaker-execution`,
      assumedBy: new iam.ServicePrincipal('sagemaker.amazonaws.com'),
      managedPolicies: [
        iam.ManagedPolicy.fromAwsManagedPolicyName('AmazonSageMakerFullAccess'),
      ],
    });

    // Grant access to model artifacts in S3
    this.sagemakerRole.addToPolicy(new iam.PolicyStatement({
      actions: [
        's3:GetObject',
        's3:ListBucket',
      ],
      resources: [
        'arn:aws:s3:::jumpstart-cache-prod-*',
        'arn:aws:s3:::jumpstart-cache-prod-*/*',
```

```
        'arn:aws:s3:::huggingface-sagemaker-*',
        'arn:aws:s3:::huggingface-sagemaker-*/*',
      ],
    }));

    // Grant CloudWatch Logs access
    this.sagemakerRole.addToPolicy(new iam.PolicyStatement({
      actions: [
        'logs:CreateLogGroup',
        'logs:CreateLogStream',
        'logs:PutLogEvents',
      ],
      resources: [`arn:aws:logs:${this.region}:${this.account}:log-group:/aws/sagemaker/*`
    }));

    // Create self-hosted model endpoints based on tier
    this.createSageMakerEndpoints(resourcePrefix, tierConfig);
  }

  // ============================================================================
  // SSM PARAMETERS
  // ============================================================================

  new ssm.StringParameter(this, 'LitellmUrlParam', {
    parameterName: `/radiant/${props.appId}/${props.environment}/ai/litellm-url`,
    stringValue: this.litellmUrl,
  });

  new ssm.StringParameter(this, 'LitellmAlbDnsParam', {
    parameterName: `/radiant/${props.appId}/${props.environment}/ai/litellm-alb-dns`,
    stringValue: this.litellmService.loadBalancer.loadBalancerDnsName,
  });

  new ssm.StringParameter(this, 'ProviderSecretsArnParam', {
    parameterName: `/radiant/${props.appId}/${props.environment}/ai/provider-secrets-arn`,
    stringValue: this.providerSecretsArn,
  });

  new ssm.StringParameter(this, 'ClusterArnParam', {
    parameterName: `/radiant/${props.appId}/${props.environment}/ai/ecs-cluster-arn`,
    stringValue: this.cluster.clusterArn,
  });

  // ============================================================================
  // TAGS
  // ============================================================================
```

```
    applyTags(this, {
      appId: props.appId,
      environment: props.environment,
      tier: props.tier,
    });

    // ============================================================================
    // OUTPUTS
    // ============================================================================

    new cdk.CfnOutput(this, 'ClusterName', {
      value: this.cluster.clusterName,
      description: 'ECS Cluster Name',
      exportName: `${resourcePrefix}-ai-cluster-name`,
    });

    new cdk.CfnOutput(this, 'LitellmUrl', {
      value: this.litellmUrl,
      description: 'LiteLLM Internal URL',
      exportName: `${resourcePrefix}-litellm-url`,
    });

    new cdk.CfnOutput(this, 'LitellmAlbDns', {
      value: this.litellmService.loadBalancer.loadBalancerDnsName,
      description: 'LiteLLM ALB DNS Name',
      exportName: `${resourcePrefix}-litellm-alb-dns`,
    });

    new cdk.CfnOutput(this, 'ProviderSecretsArn', {
      value: this.providerSecretsArn,
      description: 'Provider Secrets ARN',
      exportName: `${resourcePrefix}-provider-secrets-arn`,
    });

    new cdk.CfnOutput(this, 'ServiceDiscoveryNamespace', {
      value: this.namespace.namespaceName,
      description: 'Service Discovery Namespace',
      exportName: `${resourcePrefix}-ai-namespace`,
    });
  }

  /**
   * Create SageMaker endpoints for self-hosted models
   */
  private createSageMakerEndpoints(resourcePrefix: string, tierConfig: TierConfig): void {
```

```
// Define model configurations based on tier
const modelConfigs = this.getModelConfigs(tierConfig);

for (const config of modelConfigs) {
  // Model
  const model = new sagemaker.CfnModel(this, `Model${config.id}`, {
    modelName: `${resourcePrefix}-${config.id}`,
    executionRoleArn: this.sagemakerRole!.roleArn,
    primaryContainer: {
      image: config.image,
      environment: config.environment,
      modelDataUrl: config.modelDataUrl,
    },
  });

  // Endpoint Config
  const endpointConfig = new sagemaker.CfnEndpointConfig(this, `EndpointConfig${config.i
    endpointConfigName: `${resourcePrefix}-${config.id}-config`,
    productionVariants: [
      {
        variantName: 'AllTraffic',
        modelName: model.modelName!,
        initialInstanceCount: 0, // Start scaled to zero
        instanceType: config.instanceType,
        initialVariantWeight: 1,
      },
    ],
  });
  endpointConfig.addDependency(model);

  // Endpoint
  const endpoint = new sagemaker.CfnEndpoint(this, `Endpoint${config.id}`, {
    endpointName: `${resourcePrefix}-${config.id}`,
    endpointConfigName: endpointConfig.endpointConfigName!,
  });
  endpoint.addDependency(endpointConfig);

  this.sagemakerEndpoints.set(config.id, endpoint);

  // Output
  new cdk.CfnOutput(this, `Endpoint${config.id}Name`, {
    value: endpoint.endpointName!,
    description: `SageMaker Endpoint for ${config.name}`,
    exportName: `${resourcePrefix}-sagemaker-${config.id}`,
  });
}
```

```
}

/**
 * Get model configurations based on tier
 */
private getModelConfigs(tierConfig: TierConfig): ModelConfig[] {
  const configs: ModelConfig[] = [];

  // Tier 3: Basic models
  if (tierConfig.tier >= 3) {
    configs.push(
      {
        id: 'mistral-7b',
        name: 'Mistral 7B Instruct',
        image: `763104351884.dkr.ecr.${this.region}.amazonaws.com/huggingface-pytorch-tgi-
        instanceType: 'ml.g4dn.xlarge',
        environment: {
          HF_MODEL_ID: 'mistralai/Mistral-7B-Instruct-v0.2',
          MAX_INPUT_LENGTH: '4096',
          MAX_TOTAL_TOKENS: '8192',
        },
      },
      {
        id: 'whisper-large',
        name: 'Whisper Large V3',
        image: `763104351884.dkr.ecr.${this.region}.amazonaws.com/huggingface-pytorch-infe
        instanceType: 'ml.g4dn.xlarge',
        environment: {
          HF_MODEL_ID: 'openai/whisper-large-v3',
          HF_TASK: 'automatic-speech-recognition',
        },
      },
    );
  }

  // Tier 4+: Advanced models
  if (tierConfig.tier >= 4) {
    configs.push(
      {
        id: 'llama-70b',
        name: 'Llama 2 70B Chat',
        image: `763104351884.dkr.ecr.${this.region}.amazonaws.com/huggingface-pytorch-tgi-
        instanceType: 'ml.g5.12xlarge',
        environment: {
          HF_MODEL_ID: 'meta-llama/Llama-2-70b-chat-hf',
          MAX_INPUT_LENGTH: '4096',
```

```
        MAX_TOTAL_TOKENS: '8192',
        SM_NUM_GPUS: '4',
      },
    },
    {
      id: 'sdxl',
      name: 'Stable Diffusion XL',
      image: `763104351884.dkr.ecr.${this.region}.amazonaws.com/stabilityai-pytorch-infe
      instanceType: 'ml.g5.2xlarge',
      environment: {
        MODEL_ID: 'stabilityai/stable-diffusion-xl-base-1.0',
      },
    },
    {
      id: 'yolov8',
      name: 'YOLOv8 Object Detection',
      image: `763104351884.dkr.ecr.${this.region}.amazonaws.com/pytorch-inference:2.1-gp
      instanceType: 'ml.g4dn.xlarge',
      environment: {
        MODEL_NAME: 'yolov8x',
      },
    },
  );
}

// Tier 5: Enterprise models
if (tierConfig.tier >= 5) {
  configs.push(
    {
      id: 'qwen-72b',
      name: 'Qwen 72B Chat',
      image: `763104351884.dkr.ecr.${this.region}.amazonaws.com/huggingface-pytorch-tgi-
      instanceType: 'ml.p4d.24xlarge',
      environment: {
        HF_MODEL_ID: 'Qwen/Qwen-72B-Chat',
        MAX_INPUT_LENGTH: '8192',
        MAX_TOTAL_TOKENS: '16384',
        SM_NUM_GPUS: '8',
      },
    },
    {
      id: 'esm2',
      name: 'ESM-2 Protein',
      image: `763104351884.dkr.ecr.${this.region}.amazonaws.com/huggingface-pytorch-infe
      instanceType: 'ml.g5.4xlarge',
      environment: {
```

```
            HF_MODEL_ID: 'facebook/esm2_t36_3B_UR50D',
            HF_TASK: 'feature-extraction',
          },
        },
      );
    }

    return configs;
  }
}

interface ModelConfig {
  id: string;
  name: string;
  image: string;
  instanceType: string;
  environment: Record<string, string>;
  modelDataUrl?: string;
}
```

PART 4: API STACK (REST + GRAPHQL)

packages/infrastructure/lib/stacks/api.stack.ts

```
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as apigateway from 'aws-cdk-lib/aws-apigateway';
import * as appsync from 'aws-cdk-lib/aws-appsync';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as lambdaNodejs from 'aws-cdk-lib/aws-lambda-nodejs';
import * as cognito from 'aws-cdk-lib/aws-cognito';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as rds from 'aws-cdk-lib/aws-rds';
import * as s3 from 'aws-cdk-lib/aws-s3';
import * as iam from 'aws-cdk-lib/aws-iam';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import * as logs from 'aws-cdk-lib/aws-logs';
import * as wafv2 from 'aws-cdk-lib/aws-wafv2';
import * as secretsmanager from 'aws-cdk-lib/aws-secretsmanager';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface APIStackProps extends cdk.StackProps {
  appId: string;
```

```
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
  vpc: ec2.Vpc;
  lambdaSecurityGroup: ec2.SecurityGroup;
  userPool: cognito.UserPool;
  userPoolClient: cognito.UserPoolClient;
  litellmUrl: string;
  auroraCluster: rds.DatabaseCluster;
  auroraSecret: secretsmanager.ISecret;
  usageTable: dynamodb.Table;
  sessionsTable: dynamodb.Table;
  cacheTable: dynamodb.Table;
  mediaBucket: s3.Bucket;
  lambdaRole: iam.Role;
  webAcl?: wafv2.CfnWebACL;
}

/**
 * API Stack
 *
 * Creates REST API Gateway for external integrations and
 * AppSync GraphQL API for client applications.
 */
export class APIStack extends cdk.Stack {
  // REST API
  public readonly api: apigateway.RestApi;
  public readonly apiUrl: string;

  // GraphQL API
  public readonly graphqlApi: appsync.GraphqlApi;
  public readonly graphqlUrl: string;

  // Lambda Functions
  public readonly routerFunction: lambda.Function;
  public readonly chatFunction: lambda.Function;
  public readonly modelsFunction: lambda.Function;
  public readonly providersFunction: lambda.Function;

  constructor(scope: Construct, id: string, props: APIStackProps) {
    super(scope, id, props);

    const { tierConfig, vpc } = props;
    const resourcePrefix = `${props.appId}-${props.environment}`;
```

```
// ============================================================================
// SHARED LAMBDA CONFIGURATION
// ============================================================================

const lambdaEnvironment = {
  APP_ID: props.appId,
  ENVIRONMENT: props.environment,
  TIER: String(props.tier),
  LITELLM_URL: props.litellmUrl,
  AURORA_SECRET_ARN: props.auroraSecret.secretArn,
  AURORA_CLUSTER_ARN: props.auroraCluster.clusterArn,
  USAGE_TABLE: props.usageTable.tableName,
  SESSIONS_TABLE: props.sessionsTable.tableName,
  CACHE_TABLE: props.cacheTable.tableName,
  MEDIA_BUCKET: props.mediaBucket.bucketName,
  USER_POOL_ID: props.userPool.userPoolId,
  LOG_LEVEL: props.environment === 'prod' ? 'info' : 'debug',
};

const lambdaLogGroup = new logs.LogGroup(this, 'LambdaLogs', {
  logGroupName: `/radiant/${props.appId}/${props.environment}/lambda`,
  retention: tierConfig.tier >= 3
    ? logs.RetentionDays.THREE_MONTHS
    : logs.RetentionDays.TWO_WEEKS,
  removalPolicy: cdk.RemovalPolicy.DESTROY,
});

// ============================================================================
// LAMBDA FUNCTIONS
// ============================================================================

// Router Lambda (main entry point)
this.routerFunction = new lambdaNodejs.NodejsFunction(this, 'RouterFunction', {
  functionName: `${resourcePrefix}-router`,
  runtime: lambda.Runtime.NODEJS_20_X,
  handler: 'handler',
  entry: 'lambda/api/router.ts',
  timeout: cdk.Duration.seconds(tierConfig.lambda.timeoutSeconds),
  memorySize: tierConfig.lambda.memoryMB,
  vpc,
  vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
  securityGroups: [props.lambdaSecurityGroup],
  environment: lambdaEnvironment,
  role: props.lambdaRole,
  tracing: tierConfig.features.xrayTracing
```

```
    ? lambda.Tracing.ACTIVE
    : lambda.Tracing.DISABLED,
  logGroup: lambdaLogGroup,
  bundling: {
    minify: true,
    sourceMap: true,
    target: 'node20',
    externalModules: ['@aws-sdk/*'],
  },
});

// Chat Lambda
this.chatFunction = new lambdaNodejs.NodejsFunction(this, 'ChatFunction', {
  functionName: `${resourcePrefix}-chat`,
  runtime: lambda.Runtime.NODEJS_20_X,
  handler: 'handler',
  entry: 'lambda/api/chat.ts',
  timeout: cdk.Duration.seconds(300), // Longer timeout for AI requests
  memorySize: tierConfig.lambda.memoryMB,
  vpc,
  vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
  securityGroups: [props.lambdaSecurityGroup],
  environment: lambdaEnvironment,
  role: props.lambdaRole,
  tracing: tierConfig.features.xrayTracing
    ? lambda.Tracing.ACTIVE
    : lambda.Tracing.DISABLED,
  logGroup: lambdaLogGroup,
  bundling: {
    minify: true,
    sourceMap: true,
    target: 'node20',
    externalModules: ['@aws-sdk/*'],
  },
});

// Models Lambda
this.modelsFunction = new lambdaNodejs.NodejsFunction(this, 'ModelsFunction', {
  functionName: `${resourcePrefix}-models`,
  runtime: lambda.Runtime.NODEJS_20_X,
  handler: 'handler',
  entry: 'lambda/api/models.ts',
  timeout: cdk.Duration.seconds(30),
  memorySize: 512,
  vpc,
  vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
```

```
    securityGroups: [props.lambdaSecurityGroup],
    environment: lambdaEnvironment,
    role: props.lambdaRole,
    logGroup: lambdaLogGroup,
    bundling: {
      minify: true,
      sourceMap: true,
      target: 'node20',
      externalModules: ['@aws-sdk/*'],
    },
  });

  // Providers Lambda
  this.providersFunction = new lambdaNodejs.NodejsFunction(this, 'ProvidersFunction', {
    functionName: `${resourcePrefix}-providers`,
    runtime: lambda.Runtime.NODEJS_20_X,
    handler: 'handler',
    entry: 'lambda/api/providers.ts',
    timeout: cdk.Duration.seconds(30),
    memorySize: 512,
    vpc,
    vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
    securityGroups: [props.lambdaSecurityGroup],
    environment: lambdaEnvironment,
    role: props.lambdaRole,
    logGroup: lambdaLogGroup,
    bundling: {
      minify: true,
      sourceMap: true,
      target: 'node20',
      externalModules: ['@aws-sdk/*'],
    },
  });

  // Grant permissions
  props.auroraSecret.grantRead(this.routerFunction);
  props.auroraSecret.grantRead(this.chatFunction);
  props.auroraSecret.grantRead(this.modelsFunction);
  props.auroraSecret.grantRead(this.providersFunction);

  props.usageTable.grantReadWriteData(this.chatFunction);
  props.sessionsTable.grantReadWriteData(this.chatFunction);
  props.cacheTable.grantReadWriteData(this.routerFunction);

  props.mediaBucket.grantReadWrite(this.chatFunction);
```

```
// ============================================================================
// REST API GATEWAY
// ============================================================================

this.api = new apigateway.RestApi(this, 'RestApi', {
  restApiName: `${resourcePrefix}-api`,
  description: `RADIANT REST API for ${props.appName}`,

  deployOptions: {
    stageName: props.environment,
    tracingEnabled: tierConfig.features.xrayTracing,
    loggingLevel: apigateway.MethodLoggingLevel.INFO,
    dataTraceEnabled: props.environment !== 'prod',
    metricsEnabled: true,
    throttlingBurstLimit: tierConfig.tier >= 4 ? 5000 : 1000,
    throttlingRateLimit: tierConfig.tier >= 4 ? 10000 : 2000,
  },

  // CORS
  defaultCorsPreflightOptions: {
    allowOrigins: [
      `https://${props.domain}`,
      `https://*.${props.domain}`,
      'http://localhost:*',
    ],
    allowMethods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
    allowHeaders: [
      'Content-Type',
      'Authorization',
      'X-Api-Key',
      'X-Tenant-Id',
    ],
    allowCredentials: true,
    maxAge: cdk.Duration.hours(1),
  },

  // Binary media types
  binaryMediaTypes: [
    'image/*',
    'audio/*',
    'video/*',
    'application/octet-stream',
  ],

  // Minimum compression
  minimumCompressionSize: 1024,
```

```
  // Endpoint configuration
  endpointConfiguration: {
    types: [apigateway.EndpointType.REGIONAL],
  },
});

this.apiUrl = this.api.url;

// Cognito Authorizer
const cognitoAuthorizer = new apigateway.CognitoUserPoolsAuthorizer(this, 'CognitoAutho
  cognitoUserPools: [props.userPool],
  authorizerName: `${resourcePrefix}-cognito-auth`,
  identitySource: 'method.request.header.Authorization',
});

// API Routes
const v2 = this.api.root.addResource('api').addResource('v2');

// /api/v2/chat
const chat = v2.addResource('chat');
chat.addResource('completions').addMethod('POST',
  new apigateway.LambdaIntegration(this.chatFunction),
  {
    authorizer: cognitoAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

// /api/v2/models
const models = v2.addResource('models');
models.addMethod('GET',
  new apigateway.LambdaIntegration(this.modelsFunction),
  {
    authorizer: cognitoAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
models.addResource('{modelId}').addMethod('GET',
  new apigateway.LambdaIntegration(this.modelsFunction),
  {
    authorizer: cognitoAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
```

```
// /api/v2/providers
const providers = v2.addResource('providers');
providers.addMethod('GET',
  new apigateway.LambdaIntegration(this.providersFunction),
  {
    authorizer: cognitoAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

// /api/v2/health (no auth)
v2.addResource('health').addMethod('GET',
  new apigateway.LambdaIntegration(this.routerFunction),
  {
    authorizationType: apigateway.AuthorizationType.NONE,
  }
);

// Associate WAF (Tier 2+)
if (props.webAcl) {
  new wafv2.CfnWebACLAssociation(this, 'ApiWafAssociation', {
    resourceArn: this.api.deploymentStage.stageArn,
    webAclArn: props.webAcl.attrArn,
  });
}

// ============================================================================
// APPSYNC GRAPHQL API
// ============================================================================

this.graphqlApi = new appsync.GraphqlApi(this, 'GraphqlApi', {
  name: `${resourcePrefix}-graphql`,

  // Schema
  definition: appsync.Definition.fromFile('graphql/schema.graphql'),

  // Authorization
  authorizationConfig: {
    defaultAuthorization: {
      authorizationType: appsync.AuthorizationType.USER_POOL,
      userPoolConfig: {
        userPool: props.userPool,
        defaultAction: appsync.UserPoolDefaultAction.ALLOW,
      },
    },
    additionalAuthorizationModes: [
```

```
      {
        authorizationType: appsync.AuthorizationType.API_KEY,
        apiKeyConfig: {
          name: 'default',
          expires: cdk.Expiration.after(cdk.Duration.days(365)),
        },
      },
      {
        authorizationType: appsync.AuthorizationType.IAM,
      },
    ],
  },

  // Logging
  logConfig: {
    fieldLogLevel: props.environment === 'prod'
      ? appsync.FieldLogLevel.ERROR
      : appsync.FieldLogLevel.ALL,
    excludeVerboseContent: props.environment === 'prod',
  },

  // X-Ray
  xrayEnabled: tierConfig.features.xrayTracing,
});

this.graphqlUrl = this.graphqlApi.graphqlUrl;

// Lambda Data Source
const lambdaDs = this.graphqlApi.addLambdaDataSource(
  'LambdaDataSource',
  this.routerFunction,
  { name: 'LambdaResolver' }
);

// DynamoDB Data Source
const dynamoDs = this.graphqlApi.addDynamoDbDataSource(
  'DynamoDataSource',
  props.sessionsTable,
  { name: 'DynamoResolver' }
);

// Resolvers
lambdaDs.createResolver('QueryModels', {
  typeName: 'Query',
  fieldName: 'models',
});
```

```
lambdaDs.createResolver('QueryModel', {
  typeName: 'Query',
  fieldName: 'model',
});

lambdaDs.createResolver('QueryProviders', {
  typeName: 'Query',
  fieldName: 'providers',
});

lambdaDs.createResolver('MutationChat', {
  typeName: 'Mutation',
  fieldName: 'chat',
});

dynamoDs.createResolver('QuerySessions', {
  typeName: 'Query',
  fieldName: 'sessions',
  requestMappingTemplate: appsync.MappingTemplate.dynamoDbQuery(
    appsync.KeyCondition.eq('userId', 'userId'),
    'gsi1'
  ),
  responseMappingTemplate: appsync.MappingTemplate.dynamoDbResultList(),
});

// ============================================================================
// SSM PARAMETERS
// ============================================================================

new ssm.StringParameter(this, 'ApiUrlParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/api/rest-url`,
  stringValue: this.apiUrl,
});

new ssm.StringParameter(this, 'GraphqlUrlParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/api/graphql-url`,
  stringValue: this.graphqlUrl,
});

new ssm.StringParameter(this, 'ApiIdParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/api/rest-api-id`,
  stringValue: this.api.restApiId,
});

// ============================================================================
```

```
    // TAGS
    // ============================================================================

    applyTags(this, {
      appId: props.appId,
      environment: props.environment,
      tier: props.tier,
    });

    // ============================================================================
    // OUTPUTS
    // ============================================================================

    new cdk.CfnOutput(this, 'RestApiUrl', {
      value: this.apiUrl,
      description: 'REST API URL',
      exportName: `${resourcePrefix}-api-url`,
    });

    new cdk.CfnOutput(this, 'RestApiId', {
      value: this.api.restApiId,
      description: 'REST API ID',
      exportName: `${resourcePrefix}-api-id`,
    });

    new cdk.CfnOutput(this, 'GraphqlApiUrl', {
      value: this.graphqlUrl,
      description: 'GraphQL API URL',
      exportName: `${resourcePrefix}-graphql-url`,
    });

    new cdk.CfnOutput(this, 'GraphqlApiId', {
      value: this.graphqlApi.apiId,
      description: 'GraphQL API ID',
      exportName: `${resourcePrefix}-graphql-api-id`,
    });

    new cdk.CfnOutput(this, 'GraphqlApiKey', {
      value: this.graphqlApi.apiKey || '',
      description: 'GraphQL API Key (for development)',
      exportName: `${resourcePrefix}-graphql-api-key`,
    });
  }
}
```

PART 5: ADMIN STACK

packages/infrastructure/lib/stacks/admin.stack.ts

```
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as s3 from 'aws-cdk-lib/aws-s3';
import * as cloudfront from 'aws-cdk-lib/aws-cloudfront';
import * as origins from 'aws-cdk-lib/aws-cloudfront-origins';
import * as cognito from 'aws-cdk-lib/aws-cognito';
import * as apigateway from 'aws-cdk-lib/aws-apigateway';
import * as appsync from 'aws-cdk-lib/aws-appsync';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as lambdaNodejs from 'aws-cdk-lib/aws-lambda-nodejs';
import * as iam from 'aws-cdk-lib/aws-iam';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import * as logs from 'aws-cdk-lib/aws-logs';
import * as route53 from 'aws-cdk-lib/aws-route53';
import * as route53targets from 'aws-cdk-lib/aws-route53-targets';
import * as acm from 'aws-cdk-lib/aws-certificatemanager';
import { Construct } from 'constructs';
import { TierConfig, TierLevel } from '../config/tiers';
import { applyTags } from '../config/tags';

export interface AdminStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  tier: TierLevel;
  tierConfig: TierConfig;
  domain: string;
  vpc: ec2.Vpc;
  adminUserPool: cognito.UserPool;
  adminUserPoolClient: cognito.UserPoolClient;
  restApi: apigateway.RestApi;
  graphqlApi: appsync.GraphqlApi;
  assetsBucket: s3.Bucket;
  assetsDistribution: cloudfront.Distribution;
  hostedZoneId?: string;
}

/**
 * Admin Stack
 *
 * Creates admin dashboard hosting and admin-specific APIs.
 * Includes invitation system, two-person approval, and audit logging.
```

```
 */
export class AdminStack extends cdk.Stack {
  // Admin Dashboard
  public readonly adminBucket: s3.Bucket;
  public readonly adminDistribution: cloudfront.Distribution;
  public readonly adminUrl: string;

  // Admin Lambda Functions
  public readonly adminFunction: lambda.Function;
  public readonly invitationFunction: lambda.Function;
  public readonly approvalFunction: lambda.Function;

  constructor(scope: Construct, id: string, props: AdminStackProps) {
    super(scope, id, props);

    const { tierConfig, vpc } = props;
    const resourcePrefix = `${props.appId}-${props.environment}`;
    const adminDomain = `admin.${props.domain}`;

    // ============================================================================
    // ADMIN DASHBOARD S3 BUCKET
    // ============================================================================

    this.adminBucket = new s3.Bucket(this, 'AdminBucket', {
      bucketName: `${resourcePrefix}-admin-${this.account}-${this.region}`,

      encryption: s3.BucketEncryption.S3_MANAGED,
      blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
      enforceSSL: true,

      versioned: true,

      removalPolicy: cdk.RemovalPolicy.DESTROY,
      autoDeleteObjects: true,
    });

    // ============================================================================
    // CLOUDFRONT DISTRIBUTION FOR ADMIN DASHBOARD
    // ============================================================================

    const adminOai = new cloudfront.OriginAccessIdentity(this, 'AdminOai', {
      comment: `OAI for ${props.appName} admin dashboard`,
    });

    this.adminBucket.grantRead(adminOai);
```

```
// Response headers policy for security
const securityHeadersPolicy = new cloudfront.ResponseHeadersPolicy(this, 'AdminSecurityH
  responseHeadersPolicyName: `${resourcePrefix}-admin-security`,
  securityHeadersBehavior: {
    contentSecurityPolicy: {
      contentSecurityPolicy: `
        default-src 'self';
        script-src 'self' 'unsafe-inline' 'unsafe-eval';
        style-src 'self' 'unsafe-inline';
        img-src 'self' data: https:;
        font-src 'self' data:;
        connect-src 'self' https://*.amazonaws.com https://*.amazoncognito.com;
        frame-ancestors 'none';
      `.replace(/\s+/g, ' ').trim(),
      override: true,
    },
    contentTypeOptions: { override: true },
    frameOptions: {
      frameOption: cloudfront.HeadersFrameOption.DENY,
      override: true,
    },
    referrerPolicy: {
      referrerPolicy: cloudfront.HeadersReferrerPolicy.STRICT_ORIGIN_WHEN_CROSS_ORIGIN,
      override: true,
    },
    strictTransportSecurity: {
      accessControlMaxAge: cdk.Duration.days(365),
      includeSubdomains: true,
      preload: true,
      override: true,
    },
    xssProtection: {
      protection: true,
      modeBlock: true,
      override: true,
    },
  },
});

// Cache policy for static assets
const adminCachePolicy = new cloudfront.CachePolicy(this, 'AdminCachePolicy', {
  cachePolicyName: `${resourcePrefix}-admin-cache`,
  defaultTtl: cdk.Duration.days(1),
  minTtl: cdk.Duration.seconds(0),
  maxTtl: cdk.Duration.days(365),
  enableAcceptEncodingGzip: true,
```

```
  enableAcceptEncodingBrotli: true,
  queryStringBehavior: cloudfront.CacheQueryStringBehavior.none(),
});

this.adminDistribution = new cloudfront.Distribution(this, 'AdminDistribution', {
  comment: `${props.appName} Admin Dashboard`,

  defaultBehavior: {
    origin: new origins.S3Origin(this.adminBucket, {
      originAccessIdentity: adminOai,
    }),
    viewerProtocolPolicy: cloudfront.ViewerProtocolPolicy.REDIRECT_TO_HTTPS,
    allowedMethods: cloudfront.AllowedMethods.ALLOW_GET_HEAD,
    cachedMethods: cloudfront.CachedMethods.CACHE_GET_HEAD,
    cachePolicy: adminCachePolicy,
    responseHeadersPolicy: securityHeadersPolicy,
    compress: true,
  },

  // SPA routing
  defaultRootObject: 'index.html',
  errorResponses: [
    {
      httpStatus: 404,
      responseHttpStatus: 200,
      responsePagePath: '/index.html',
      ttl: cdk.Duration.seconds(0),
    },
    {
      httpStatus: 403,
      responseHttpStatus: 200,
      responsePagePath: '/index.html',
      ttl: cdk.Duration.seconds(0),
    },
  ],

  priceClass: cloudfront.PriceClass.PRICE_CLASS_100,
  httpVersion: cloudfront.HttpVersion.HTTP2_AND_3,
  minimumProtocolVersion: cloudfront.SecurityPolicyProtocol.TLS_V1_2_2021,
});

this.adminUrl = `https://${this.adminDistribution.distributionDomainName}`;

// ============================================================================
// ADMIN LAMBDA FUNCTIONS
// ============================================================================
```

```
const adminLogGroup = new logs.LogGroup(this, 'AdminLambdaLogs', {
  logGroupName: `/radiant/${props.appId}/${props.environment}/admin-lambda`,
  retention: tierConfig.tier >= 3
    ? logs.RetentionDays.ONE_YEAR
    : logs.RetentionDays.THREE_MONTHS,
  removalPolicy: cdk.RemovalPolicy.DESTROY,
});

const adminLambdaRole = new iam.Role(this, 'AdminLambdaRole', {
  roleName: `${resourcePrefix}-admin-lambda`,
  assumedBy: new iam.ServicePrincipal('lambda.amazonaws.com'),
  managedPolicies: [
    iam.ManagedPolicy.fromAwsManagedPolicyName('service-role/AWSLambdaVPCAccessExecutio
  ],
});

// Cognito admin permissions
adminLambdaRole.addToPolicy(new iam.PolicyStatement({
  actions: [
    'cognito-idp:AdminCreateUser',
    'cognito-idp:AdminDeleteUser',
    'cognito-idp:AdminGetUser',
    'cognito-idp:AdminUpdateUserAttributes',
    'cognito-idp:AdminListGroupsForUser',
    'cognito-idp:AdminAddUserToGroup',
    'cognito-idp:AdminRemoveUserFromGroup',
    'cognito-idp:ListUsers',
    'cognito-idp:ListUsersInGroup',
  ],
  resources: [props.adminUserPool.userPoolArn],
}));

// SSM permissions
adminLambdaRole.addToPolicy(new iam.PolicyStatement({
  actions: ['ssm:GetParameter', 'ssm:GetParameters'],
  resources: [`arn:aws:ssm:${this.region}:${this.account}:parameter/radiant/${props.appl
}));

const adminEnvironment = {
  APP_ID: props.appId,
  ENVIRONMENT: props.environment,
  ADMIN_USER_POOL_ID: props.adminUserPool.userPoolId,
  ADMIN_CLIENT_ID: props.adminUserPoolClient.userPoolClientId,
  ADMIN_URL: this.adminUrl,
  LOG_LEVEL: props.environment === 'prod' ? 'info' : 'debug',
```

```
};

// Admin CRUD Function
this.adminFunction = new lambdaNodejs.NodejsFunction(this, 'AdminFunction', {
  functionName: `${resourcePrefix}-admin-api`,
  runtime: lambda.Runtime.NODEJS_20_X,
  handler: 'handler',
  entry: 'lambda/admin/admin.ts',
  timeout: cdk.Duration.seconds(30),
  memorySize: 512,
  vpc,
  vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
  environment: adminEnvironment,
  role: adminLambdaRole,
  logGroup: adminLogGroup,
  bundling: {
    minify: true,
    sourceMap: true,
    target: 'node20',
    externalModules: ['@aws-sdk/*'],
  },
});

// Invitation Function
this.invitationFunction = new lambdaNodejs.NodejsFunction(this, 'InvitationFunction', {
  functionName: `${resourcePrefix}-invitation`,
  runtime: lambda.Runtime.NODEJS_20_X,
  handler: 'handler',
  entry: 'lambda/admin/invitation.ts',
  timeout: cdk.Duration.seconds(30),
  memorySize: 512,
  vpc,
  vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
  environment: {
    ...adminEnvironment,
    SES_SENDER: `noreply@${props.domain}`,
  },
  role: adminLambdaRole,
  logGroup: adminLogGroup,
  bundling: {
    minify: true,
    sourceMap: true,
    target: 'node20',
    externalModules: ['@aws-sdk/*'],
  },
});
```

```
// Add SES permissions for invitations
this.invitationFunction.addToRolePolicy(new iam.PolicyStatement({
  actions: ['ses:SendEmail', 'ses:SendRawEmail'],
  resources: ['*'],
}));

// Two-Person Approval Function
this.approvalFunction = new lambdaNodejs.NodejsFunction(this, 'ApprovalFunction', {
  functionName: `${resourcePrefix}-approval`,
  runtime: lambda.Runtime.NODEJS_20_X,
  handler: 'handler',
  entry: 'lambda/admin/approval.ts',
  timeout: cdk.Duration.seconds(60),
  memorySize: 512,
  vpc,
  vpcSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_EGRESS },
  environment: adminEnvironment,
  role: adminLambdaRole,
  logGroup: adminLogGroup,
  bundling: {
    minify: true,
    sourceMap: true,
    target: 'node20',
    externalModules: ['@aws-sdk/*'],
  },
});

// ============================================================================
// ADMIN API ROUTES
// ============================================================================

// Add admin routes to existing REST API
const adminAuthorizer = new apigateway.CognitoUserPoolsAuthorizer(this, 'AdminAuthorizer
  cognitoUserPools: [props.adminUserPool],
  authorizerName: `${resourcePrefix}-admin-auth`,
});

const adminApi = props.restApi.root.addResource('admin');

// /admin/users
const users = adminApi.addResource('users');
users.addMethod('GET',
  new apigateway.LambdaIntegration(this.adminFunction),
  {
    authorizer: adminAuthorizer,
```

```
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
users.addMethod('POST',
  new apigateway.LambdaIntegration(this.adminFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

const userById = users.addResource('{userId}');
userById.addMethod('GET',
  new apigateway.LambdaIntegration(this.adminFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
userById.addMethod('PUT',
  new apigateway.LambdaIntegration(this.adminFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
userById.addMethod('DELETE',
  new apigateway.LambdaIntegration(this.adminFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

// /admin/invitations
const invitations = adminApi.addResource('invitations');
invitations.addMethod('GET',
  new apigateway.LambdaIntegration(this.invitationFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
invitations.addMethod('POST',
  new apigateway.LambdaIntegration(this.invitationFunction),
  {
```

```
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

const invitationById = invitations.addResource('{invitationId}');
invitationById.addMethod('DELETE',
  new apigateway.LambdaIntegration(this.invitationFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
invitationById.addResource('resend').addMethod('POST',
  new apigateway.LambdaIntegration(this.invitationFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

// /admin/promotions
const promotions = adminApi.addResource('promotions');
promotions.addMethod('GET',
  new apigateway.LambdaIntegration(this.approvalFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
promotions.addMethod('POST',
  new apigateway.LambdaIntegration(this.approvalFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

const promotionById = promotions.addResource('{promotionId}');
promotionById.addResource('approve').addMethod('POST',
  new apigateway.LambdaIntegration(this.approvalFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);
```

```
promotionById.addResource('reject').addMethod('POST',
  new apigateway.LambdaIntegration(this.approvalFunction),
  {
    authorizer: adminAuthorizer,
    authorizationType: apigateway.AuthorizationType.COGNITO,
  }
);

// ============================================================================
// SSM PARAMETERS
// ============================================================================

new ssm.StringParameter(this, 'AdminUrlParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/admin/url`,
  stringValue: this.adminUrl,
});

new ssm.StringParameter(this, 'AdminBucketParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/admin/bucket`,
  stringValue: this.adminBucket.bucketName,
});

new ssm.StringParameter(this, 'AdminDistributionIdParam', {
  parameterName: `/radiant/${props.appId}/${props.environment}/admin/distribution-id`,
  stringValue: this.adminDistribution.distributionId,
});

// ============================================================================
// TAGS
// ============================================================================

applyTags(this, {
  appId: props.appId,
  environment: props.environment,
  tier: props.tier,
});

// ============================================================================
// OUTPUTS
// ============================================================================

new cdk.CfnOutput(this, 'AdminUrl', {
  value: this.adminUrl,
  description: 'Admin Dashboard URL',
  exportName: `${resourcePrefix}-admin-url`,
});
```

```
    new cdk.CfnOutput(this, 'AdminBucketName', {
      value: this.adminBucket.bucketName,
      description: 'Admin S3 Bucket',
      exportName: `${resourcePrefix}-admin-bucket`,
    });

    new cdk.CfnOutput(this, 'AdminDistributionId', {
      value: this.adminDistribution.distributionId,
      description: 'Admin CloudFront Distribution ID',
      exportName: `${resourcePrefix}-admin-distribution-id`,
    });
  }
}
```

PART 6: GRAPHQL SCHEMA

graphql/schema.graphql

```
# RADIANT GraphQL Schema v2.2.0

# ============================================================================
# DIRECTIVES
# ============================================================================

directive @auth(
  rules: [AuthRule!]!
) on OBJECT | FIELD_DEFINITION

input AuthRule {
  allow: AuthStrategy!
  groups: [String]
  operations: [ModelOperation]
}

enum AuthStrategy {
  owner
  groups
  private
  public
}

enum ModelOperation {
  create
  read
```

```
  update
  delete
}

# ============================================================================
# TYPES
# ============================================================================

type Query {
  # Models
  models(
    specialty: ModelSpecialty
    provider: String
    status: ModelStatus
    limit: Int
    nextToken: String
  ): ModelConnection!

  model(id: ID!): Model

  # Providers
  providers(
    type: ProviderType
    status: ProviderStatus
    limit: Int
    nextToken: String
  ): ProviderConnection!

  provider(id: ID!): Provider

  # Sessions
  sessions(
    userId: ID!
    limit: Int
    nextToken: String
  ): SessionConnection!

  session(id: ID!): Session

  # Usage
  usage(
    tenantId: ID!
    startDate: AWSDateTime!
    endDate: AWSDateTime!
  ): UsageReport!
}
```

```
type Mutation {
  # Chat
  chat(input: ChatInput!): ChatResponse!

  # Sessions
  createSession(input: CreateSessionInput!): Session!
  updateSession(input: UpdateSessionInput!): Session!
  deleteSession(id: ID!): Boolean!

  # Admin (requires admin group)
  updateProvider(input: UpdateProviderInput!): Provider! @auth(rules: [{ allow: groups, grou
  updateModel(input: UpdateModelInput!): Model! @auth(rules: [{ allow: groups, groups: ["adm
}

type Subscription {
  onChatResponse(sessionId: ID!): ChatStreamResponse
    @aws_subscribe(mutations: ["chat"])
}

# ============================================================================
# MODEL TYPES
# ============================================================================

type Model {
  id: ID!
  providerId: ID!
  provider: Provider
  name: String!
  displayName: String!
  description: String
  type: ModelType!
  specialty: ModelSpecialty!
  capabilities: [String!]!
  contextWindow: Int
  maxOutputTokens: Int
  supportsFunctions: Boolean
  supportsVision: Boolean
  supportsStreaming: Boolean
  hasThinkingMode: Boolean
  thinkingBudgetTokens: Int
  pricing: ModelPricing!
  thermalState: ThermalState
  status: ModelStatus!
  releaseDate: AWSDateTime
  deprecationDate: AWSDateTime
```

```
  createdAt: AWSDateTime!
  updatedAt: AWSDateTime!
}

type ModelConnection {
  items: [Model!]!
  nextToken: String
  totalCount: Int
}

type ModelPricing {
  inputTokens: Float
  outputTokens: Float
  perImage: Float
  perMinuteAudio: Float
  perMinuteVideo: Float
  per3DModel: Float
  billedMarkup: Float!
}

enum ModelType {
  EXTERNAL
  SELF_HOSTED
}

enum ModelSpecialty {
  TEXT_GENERATION
  TEXT_REASONING
  IMAGE_UNDERSTANDING
  IMAGE_GENERATION
  VIDEO_GENERATION
  AUDIO_TRANSCRIPTION
  AUDIO_GENERATION
  TEXT_TO_SPEECH
  CODE_GENERATION
  EMBEDDINGS
  THREE_D_GENERATION
  COMPUTER_VISION
  SCIENTIFIC
  MEDICAL
  GEOSPATIAL
}

enum ModelStatus {
  ACTIVE
  INACTIVE
```

```
  DEPRECATED
  COMING_SOON
}

enum ThermalState {
  OFF
  COLD
  WARM
  HOT
  AUTOMATIC
}

# ============================================================================
# PROVIDER TYPES
# ============================================================================

type Provider {
  id: ID!
  name: String!
  type: ProviderType!
  status: ProviderStatus!
  hipaaCompliant: Boolean!
  baaAvailable: Boolean!
  baseUrl: String
  authType: AuthType!
  capabilities: [String!]!
  models: [Model!]
  createdAt: AWSDateTime!
  updatedAt: AWSDateTime!
}

type ProviderConnection {
  items: [Provider!]!
  nextToken: String
  totalCount: Int
}

enum ProviderType {
  EXTERNAL
  SELF_HOSTED
  MID_TIER
}

enum ProviderStatus {
  ACTIVE
  INACTIVE
```

```
  DEPRECATED
}

enum AuthType {
  API_KEY
  OAUTH
  IAM
  NONE
}

# ============================================================================
# CHAT TYPES
# ============================================================================

input ChatInput {
  sessionId: ID
  model: String!
  messages: [MessageInput!]!
  maxTokens: Int
  temperature: Float
  stream: Boolean
  enablePHI: Boolean
  phiCategories: [String!]
}

input MessageInput {
  role: MessageRole!
  content: String!
  name: String
  images: [ImageInput!]
}

input ImageInput {
  url: String
  base64: String
  mediaType: String
}

enum MessageRole {
  system
  user
  assistant
  function
}

type ChatResponse {
```

```
  id: ID!
  sessionId: ID!
  model: String!
  message: Message!
  usage: TokenUsage!
  finishReason: String
  createdAt: AWSDateTime!
}

type ChatStreamResponse {
  id: ID!
  sessionId: ID!
  delta: String
  finishReason: String
}

type Message {
  role: MessageRole!
  content: String!
  name: String
}

type TokenUsage {
  promptTokens: Int!
  completionTokens: Int!
  totalTokens: Int!
}

# ============================================================================
# SESSION TYPES
# ============================================================================

type Session {
  id: ID!
  userId: ID!
  tenantId: ID!
  title: String
  model: String!
  messages: [Message!]!
  metadata: AWSJSON
  createdAt: AWSDateTime!
  updatedAt: AWSDateTime!
}

type SessionConnection {
  items: [Session!]!
```

```
  nextToken: String
  totalCount: Int
}

input CreateSessionInput {
  title: String
  model: String!
  metadata: AWSJSON
}

input UpdateSessionInput {
  id: ID!
  title: String
  metadata: AWSJSON
}

# ============================================================================
# USAGE TYPES
# ============================================================================

type UsageReport {
  tenantId: ID!
  period: String!
  totalCost: Float!
  totalBilled: Float!
  breakdown: [UsageBreakdown!]!
}

type UsageBreakdown {
  providerId: ID!
  modelId: ID!
  requests: Int!
  inputTokens: Int!
  outputTokens: Int!
  cost: Float!
  billed: Float!
}

# ============================================================================
# ADMIN TYPES
# ============================================================================

input UpdateProviderInput {
  id: ID!
  status: ProviderStatus
  hipaaCompliant: Boolean
```

```
}

input UpdateModelInput {
  id: ID!
  status: ModelStatus
  thermalState: ThermalState
  displayName: String
  description: String
}
```

PART 7: UPDATE STACK EXPORTS

packages/infrastructure/lib/stacks/index.ts (Updated)

```typescript
// Foundation Stacks (Prompt 2)
export * from './foundation.stack';
export * from './networking.stack';
export * from './security.stack';
export * from './data.stack';
export * from './storage.stack';

// AI & API Stacks (Prompt 3)
export * from './auth.stack';
export * from './ai.stack';
export * from './api.stack';
export * from './admin.stack';
```

DEPLOYMENT COMMANDS

Deploy All Stacks

```bash
cd packages/infrastructure

# Development (Tier 1)
npx cdk deploy --all \
  --context appId=thinktank \
  --context appName="Think Tank" \
  --context environment=dev \
  --context tier=1 \
  --context domain=thinktank.YOUR_DOMAIN.com

# Staging (Tier 2)
npx cdk deploy --all \
  --context appId=thinktank \
```

```
  --context appName="Think Tank" \
  --context environment=staging \
  --context tier=2 \
  --context domain=thinktank.YOUR_DOMAIN.com

# Production (Tier 3+)
npx cdk deploy --all \
  --context appId=thinktank \
  --context appName="Think Tank" \
  --context environment=prod \
  --context tier=3 \
  --context domain=thinktank.YOUR_DOMAIN.com
```

Deploy Individual Stacks

```
# Auth Stack only
npx cdk deploy thinktank-dev-auth \
  --context appId=thinktank \
  --context environment=dev \
  --context tier=1

# AI Stack only
npx cdk deploy thinktank-dev-ai \
  --context appId=thinktank \
  --context environment=dev \
  --context tier=3

# API Stack only
npx cdk deploy thinktank-dev-api \
  --context appId=thinktank \
  --context environment=dev \
  --context tier=2

# Admin Stack only
npx cdk deploy thinktank-dev-admin \
  --context appId=thinktank \
  --context environment=dev \
  --context tier=2
```

Verify Deployment

```
# Check LiteLLM health
curl http://$(aws ssm get-parameter \
  --name /radiant/thinktank/dev/ai/litellm-alb-dns \
  --query 'Parameter.Value' --output text)/health
```

```
# Check API health
curl $(aws ssm get-parameter \
  --name /radiant/thinktank/dev/api/rest-url \
  --query 'Parameter.Value' --output text)api/v2/health

# Get Admin URL
aws ssm get-parameter \
  --name /radiant/thinktank/dev/admin/url \
  --query 'Parameter.Value' --output text
```

ESTIMATED COSTS BY TIER

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Cognito	$0	$5	$25	$100	$500
LiteLLM (ECS)	$25	$75	$200	$500	$1,500
API Gateway	$5	$25	$100	$300	$1,000
AppSync	$5	$15	$50	$150	$500
Lambda	$5	$20	$75	$250	$750
CloudFront	$5	$15	$50	$200	$500
SageMaker	$0	$0	$200	$1,000	$5,000
Prompt 3 Total	~$45	~$155	~$700	~$2,500	~$9,750

Note: Add to Prompt 2 infrastructure costs for total monthly estimate.

NEXT PROMPTS

Continue with: - **Prompt 4**: Lambda Functions - Core (Router, Chat, Models, Providers, PHI) - **Prompt 5**: Lambda Functions - Admin & Billing (Invitations, Approvals, Metering) - **Prompt 6**: Self-Hosted Models & Mid-Level Services Configuration - **Prompt 7**: External Providers & Database Schema/Migrations - **Prompt 8**: Admin Web Dashboard (Next.js) - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 3: CDK AI & API Stacks RADIANT v2.2.0 - December 2024

â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â•

END OF SECTION 3

â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â•

â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â•

SECTION-04-LAMBDA-CORE

SECTION 4: LAMBDA FUNCTIONS - CORE (v2.1.0)

â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â• â•

> **Dependencies:** Sections 0-3 **Creates:** Router, Chat, Models, Providers, PHI handlers

Type Imports

```
import { AIModel, AIProvider, ThermalState, PHIConfig, UsageEvent } from '@radiant/shared';
```

RADIANT v2.2.0 - Prompt 4: Lambda Functions - Core

> **Prompt 4 of 9** | Target Size: ~55KB | Version: 3.7.0 | December 2024

OVERVIEW

This prompt creates the core Lambda functions for the RADIANT API:

1. **Router Lambda -** Main API entry point, request routing, health checks
2. **Chat Lambda -** AI completions via LiteLLM, streaming support
3. **Models Lambda -** Dynamic model registry CRUD operations
4. **Providers Lambda -** Provider management and configuration
5. **PHI Lambda -** HIPAA-compliant PHI sanitization and re-identification

Plus shared utilities: - Database client (Aurora PostgreSQL via Data API) - LiteLLM client (HTTP client for AI routing) - Authentication helpers (JWT validation, tenant extraction) - Response helpers (standardized API responses) - Logging and tracing utilities

---

LAMBDA DIRECTORY STRUCTURE

```
packages/infrastructure/lambda/
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ tsconfig.json
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ package.json
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ shared/
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ config.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ logger.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ errors.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ response.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ auth.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ db/
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ client.ts
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ queries.ts
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ types.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ litellm/
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ client.ts
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ types.ts
Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ phi/
Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ sanitizer.ts
Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ patterns.ts
Ã¢â€â€š       Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ types.ts
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ api/
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ router.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ chat.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ models.ts
Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ providers.ts
Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ admin/
    Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ [See Prompt 5]
```

PART 1: LAMBDA CONFIGURATION

packages/infrastructure/lambda/package.json

```
{
  "name": "@radiant/lambda",
  "version": "2.2.0",
  "private": true,
  "scripts": {
    "build": "tsc",
    "clean": "rm -rf dist",
    "lint": "eslint . --ext .ts",
    "test": "jest"
  },
  "dependencies": {
    "@aws-sdk/client-dynamodb": "^3.470.0",
    "@aws-sdk/client-rds-data": "^3.470.0",
    "@aws-sdk/client-secrets-manager": "^3.470.0",
    "@aws-sdk/client-ssm": "^3.470.0",
    "@aws-sdk/client-s3": "^3.470.0",
    "@aws-sdk/client-sagemaker-runtime": "^3.470.0",
    "@aws-sdk/lib-dynamodb": "^3.470.0",
    "@aws-sdk/s3-request-presigner": "^3.470.0",
    "uuid": "^9.0.0",
    "zod": "^3.22.0"
  },
  "devDependencies": {
    "@types/aws-lambda": "^8.10.130",
    "@types/node": "^20.10.0",
    "@types/uuid": "^9.0.0",
    "typescript": "^5.3.0",
    "jest": "^29.7.0",
    "@types/jest": "^29.5.11",
    "ts-jest": "^29.1.1"
  }
}
```

packages/infrastructure/lambda/tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "NodeNext",
    "moduleResolution": "NodeNext",
    "lib": ["ES2022"],
    "outDir": "./dist",
    "rootDir": ".",
```

```
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "declaration": true,
    "declarationMap": true,
    "sourceMap": true,
    "resolveJsonModule": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true
  },
  "include": ["**/*.ts"],
  "exclude": ["node_modules", "dist"]
}
```

PART 2: SHARED UTILITIES

packages/infrastructure/lambda/shared/index.ts

```typescript
// Re-export all shared utilities
export * from './config';
export * from './logger';
export * from './errors';
export * from './response';
export * from './auth';
export * from './db';
export * from './litellm';
export * from './phi';
```

packages/infrastructure/lambda/shared/config.ts

```typescript
/**
 * Environment configuration with validation
 */

import { z } from 'zod';

const envSchema = z.object({
  APP_ID: z.string().min(1),
  ENVIRONMENT: z.enum(['dev', 'staging', 'prod']),
  TIER: z.string().transform(Number).pipe(z.number().min(1).max(5)),
  LITELLM_URL: z.string().url(),
  AURORA_SECRET_ARN: z.string().startsWith('arn:aws:secretsmanager:'),
```

```
  AURORA_CLUSTER_ARN: z.string().startsWith('arn:aws:rds:'),
  USAGE_TABLE: z.string().min(1),
  SESSIONS_TABLE: z.string().min(1),
  CACHE_TABLE: z.string().min(1),
  MEDIA_BUCKET: z.string().min(1),
  USER_POOL_ID: z.string().min(1),
  LOG_LEVEL: z.enum(['debug', 'info', 'warn', 'error']).default('info'),
  AWS_REGION: z.string().default('us-east-1'),
});

export type Config = z.infer<typeof envSchema>;

let cachedConfig: Config | null = null;

export function getConfig(): Config {
  if (cachedConfig) return cachedConfig;

  const result = envSchema.safeParse(process.env);

  if (!result.success) {
    console.error('Configuration validation failed:', result.error.flatten());
    throw new Error(`Invalid configuration: ${JSON.stringify(result.error.flatten())}`);
  }

  cachedConfig = result.data;
  return cachedConfig;
}

/**
 * Feature flags based on tier
 */
export interface FeatureFlags {
  multiRegion: boolean;
  waf: boolean;
  guardDuty: boolean;
  sagemaker: boolean;
  elasticache: boolean;
  xray: boolean;
  phiSanitization: boolean;
  advancedMetrics: boolean;
}

export function getFeatureFlags(tier: number): FeatureFlags {
  return {
    multiRegion: tier >= 4,
    waf: tier >= 2,
```

```
    guardDuty: tier >= 2,
    sagemaker: tier >= 3,
    elasticache: tier >= 2,
    xray: tier >= 2,
    phiSanitization: true, // Always available
    advancedMetrics: tier >= 3,
  };
}
```

packages/infrastructure/lambda/shared/logger.ts

```
/**
 * Structured logging with correlation IDs
 */

export type LogLevel = 'debug' | 'info' | 'warn' | 'error';

interface LogContext {
  requestId?: string;
  tenantId?: string;
  userId?: string;
  appId?: string;
  environment?: string;
  [key: string]: unknown;
}

interface LogEntry {
  timestamp: string;
  level: LogLevel;
  message: string;
  context?: LogContext;
  error?: {
    name: string;
    message: string;
    stack?: string;
  };
  duration?: number;
  [key: string]: unknown;
}

const LOG_LEVELS: Record<LogLevel, number> = {
  debug: 0,
  info: 1,
  warn: 2,
  error: 3,
};
```

```typescript
export class Logger {
  private context: LogContext;
  private minLevel: LogLevel;
  private startTime: number;

  constructor(context: LogContext = {}, minLevel?: LogLevel) {
    this.context = context;
    this.minLevel = minLevel || (process.env.LOG_LEVEL as LogLevel) || 'info';
    this.startTime = Date.now();
  }

  private shouldLog(level: LogLevel): boolean {
    return LOG_LEVELS[level] >= LOG_LEVELS[this.minLevel];
  }

  private formatEntry(level: LogLevel, message: string, extra?: Record<string, unknown>): Lo
    return {
      timestamp: new Date().toISOString(),
      level,
      message,
      context: this.context,
      duration: Date.now() - this.startTime,
      ...extra,
    };
  }

  private log(level: LogLevel, message: string, extra?: Record<string, unknown>): void {
    if (!this.shouldLog(level)) return;

    const entry = this.formatEntry(level, message, extra);
    const output = JSON.stringify(entry);

    switch (level) {
      case 'error':
        console.error(output);
        break;
      case 'warn':
        console.warn(output);
        break;
      default:
        console.log(output);
    }
  }

  debug(message: string, extra?: Record<string, unknown>): void {
```

```
    this.log('debug', message, extra);
  }

  info(message: string, extra?: Record<string, unknown>): void {
    this.log('info', message, extra);
  }

  warn(message: string, extra?: Record<string, unknown>): void {
    this.log('warn', message, extra);
  }

  error(message: string, error?: Error, extra?: Record<string, unknown>): void {
    this.log('error', message, {
      ...extra,
      error: error ? {
        name: error.name,
        message: error.message,
        stack: error.stack,
      } : undefined,
    });
  }

  child(additionalContext: LogContext): Logger {
    return new Logger(
      { ...this.context, ...additionalContext },
      this.minLevel
    );
  }

  setRequestId(requestId: string): void {
    this.context.requestId = requestId;
  }

  setTenantId(tenantId: string): void {
    this.context.tenantId = tenantId;
  }

  setUserId(userId: string): void {
    this.context.userId = userId;
  }
}

// Default logger instance
export const logger = new Logger({
  appId: process.env.APP_ID,
  environment: process.env.ENVIRONMENT,
```

```
});
```

packages/infrastructure/lambda/shared/errors.ts

```
/**
 * Custom error types for API responses
 */

export abstract class AppError extends Error {
  abstract readonly statusCode: number;
  abstract readonly code: string;
  readonly isOperational = true;

  constructor(message: string) {
    super(message);
    this.name = this.constructor.name;
    Error.captureStackTrace(this, this.constructor);
  }

  toJSON() {
    return {
      code: this.code,
      message: this.message,
      statusCode: this.statusCode,
    };
  }
}

// 400 Bad Request
export class ValidationError extends AppError {
  readonly statusCode = 400;
  readonly code = 'VALIDATION_ERROR';
  readonly details?: Record<string, string[]>;

  constructor(message: string, details?: Record<string, string[]>) {
    super(message);
    this.details = details;
  }

  toJSON() {
    return {
      ...super.toJSON(),
      details: this.details,
    };
  }
}
```

```typescript
// 401 Unauthorized
export class UnauthorizedError extends AppError {
  readonly statusCode = 401;
  readonly code = 'UNAUTHORIZED';

  constructor(message = 'Authentication required') {
    super(message);
  }
}

// 403 Forbidden
export class ForbiddenError extends AppError {
  readonly statusCode = 403;
  readonly code = 'FORBIDDEN';

  constructor(message = 'Access denied') {
    super(message);
  }
}

// 404 Not Found
export class NotFoundError extends AppError {
  readonly statusCode = 404;
  readonly code = 'NOT_FOUND';
  readonly resource?: string;

  constructor(resource?: string) {
    super(resource ? `${resource} not found` : 'Resource not found');
    this.resource = resource;
  }
}

// 409 Conflict
export class ConflictError extends AppError {
  readonly statusCode = 409;
  readonly code = 'CONFLICT';

  constructor(message: string) {
    super(message);
  }
}

// 422 Unprocessable Entity
export class UnprocessableError extends AppError {
  readonly statusCode = 422;
```

```
  readonly code = 'UNPROCESSABLE_ENTITY';

  constructor(message: string) {
    super(message);
  }
}

// 429 Too Many Requests
export class RateLimitError extends AppError {
  readonly statusCode = 429;
  readonly code = 'RATE_LIMITED';
  readonly retryAfter?: number;

  constructor(retryAfter?: number) {
    super('Rate limit exceeded');
    this.retryAfter = retryAfter;
  }
}

// 500 Internal Server Error
export class InternalError extends AppError {
  readonly statusCode = 500;
  readonly code = 'INTERNAL_ERROR';

  constructor(message = 'An unexpected error occurred') {
    super(message);
  }
}

// 502 Bad Gateway (AI provider errors)
export class ProviderError extends AppError {
  readonly statusCode = 502;
  readonly code = 'PROVIDER_ERROR';
  readonly provider?: string;

  constructor(message: string, provider?: string) {
    super(message);
    this.provider = provider;
  }
}

// 503 Service Unavailable
export class ServiceUnavailableError extends AppError {
  readonly statusCode = 503;
  readonly code = 'SERVICE_UNAVAILABLE';
```

```
  constructor(message = 'Service temporarily unavailable') {
    super(message);
  }
}

/**
 * Check if error is an operational error (expected)
 */
export function isOperationalError(error: unknown): error is AppError {
  return error instanceof AppError && error.isOperational;
}

/**
 * Convert unknown error to AppError
 */
export function toAppError(error: unknown): AppError {
  if (error instanceof AppError) {
    return error;
  }

  if (error instanceof Error) {
    return new InternalError(error.message);
  }

  return new InternalError('Unknown error occurred');
}
```

packages/infrastructure/lambda/shared/response.ts

```
/**
 * Standardized API response helpers
 */

import type { APIGatewayProxyResult } from 'aws-lambda';
import { AppError, toAppError } from './errors';
import { Logger } from './logger';

interface SuccessResponse<T> {
  success: true;
  data: T;
  meta?: ResponseMeta;
}

interface ErrorResponse {
  success: false;
  error: {
```

```
    code: string;
    message: string;
    details?: unknown;
  };
}

interface ResponseMeta {
  requestId?: string;
  pagination?: {
    page: number;
    limit: number;
    total: number;
    hasMore: boolean;
  };
  timing?: {
    duration: number;
  };
}

type ApiResponse<T> = SuccessResponse<T> | ErrorResponse;

const DEFAULT_HEADERS = {
  'Content-Type': 'application/json',
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Headers': 'Content-Type,Authorization,X-Api-Key,X-Tenant-Id',
  'Access-Control-Allow-Methods': 'GET,POST,PUT,DELETE,OPTIONS',
  'X-Content-Type-Options': 'nosniff',
  'X-Frame-Options': 'DENY',
  'Strict-Transport-Security': 'max-age=31536000; includeSubDomains',
};

/**
 * Create success response
 */
export function success<T>(
  data: T,
  statusCode = 200,
  meta?: ResponseMeta
): APIGatewayProxyResult {
  const response: SuccessResponse<T> = {
    success: true,
    data,
    meta,
  };

  return {
```

```
    statusCode,
    headers: DEFAULT_HEADERS,
    body: JSON.stringify(response),
  };
}

/**
 * Create created response (201)
 */
export function created<T>(data: T, meta?: ResponseMeta): APIGatewayProxyResult {
  return success(data, 201, meta);
}

/**
 * Create no content response (204)
 */
export function noContent(): APIGatewayProxyResult {
  return {
    statusCode: 204,
    headers: DEFAULT_HEADERS,
    body: '',
  };
}

/**
 * Create error response
 */
export function error(
  err: AppError,
  logger?: Logger
): APIGatewayProxyResult {
  if (logger) {
    if (err.statusCode >= 500) {
      logger.error('Internal error', err);
    } else {
      logger.warn('Client error', { error: err.toJSON() });
    }
  }

  const response: ErrorResponse = {
    success: false,
    error: {
      code: err.code,
      message: err.message,
      details: (err as any).details,
    },
```

```
  };

  const headers = { ...DEFAULT_HEADERS };

  if ('retryAfter' in err && err.retryAfter) {
    headers['Retry-After'] = String(err.retryAfter);
  }

  return {
    statusCode: err.statusCode,
    headers,
    body: JSON.stringify(response),
  };
}

/**
 * Handle errors uniformly
 */
export function handleError(
  err: unknown,
  logger?: Logger
): APIGatewayProxyResult {
  const appError = toAppError(err);
  return error(appError, logger);
}

/**
 * Create streaming response headers
 */
export function streamingHeaders(): Record<string, string> {
  return {
    ...DEFAULT_HEADERS,
    'Content-Type': 'text/event-stream',
    'Cache-Control': 'no-cache',
    Connection: 'keep-alive',
  };
}

/**
 * Format Server-Sent Event
 */
export function formatSSE(data: unknown, event?: string): string {
  let output = '';
  if (event) {
    output += `event: ${event}\n`;
  }
```

```
  output += `data: ${JSON.stringify(data)}\n\n`;
  return output;
}
```

packages/infrastructure/lambda/shared/auth.ts

```
/**
 * Authentication and authorization utilities
 */

import type { APIGatewayProxyEvent } from 'aws-lambda';
import { UnauthorizedError, ForbiddenError } from './errors';
import { Logger } from './logger';

/**
 * Enhanced AuthContext with app isolation (v4.6.0)
 */
export interface AuthContext {
  // Identity
  userId: string;       // Cognito sub
  appUserId: string;    // App-scoped user ID from app_users table
  tenantId: string;
  appId: string;        // Application identifier (thinktank, launchboard, etc.)
  email: string;

  // Roles & permissions
  roles: string[];
  groups: string[];
  isAdmin: boolean;
  isSuperAdmin: boolean;

  // Session
  sessionId?: string;
  tokenExpiry: number;
}

export interface TokenClaims {
  sub: string;
  email?: string;
  'cognito:username'?: string;
  'cognito:groups'?: string[];
  'custom:tenantId'?: string;
  'custom:tenant_id'?: string;
  'custom:appId'?: string;
  'custom:app_id'?: string;
  'custom:appUserId'?: string;
```

```
  'custom:app_user_id'?: string;
  'custom:role'?: string;
  iss: string;
  aud: string;
  exp: number;
  iat: number;
}

/**
 * Extract and validate authentication context from API Gateway event
 * Includes app isolation validation (v4.6.0)
 */
export function extractAuthContext(event: APIGatewayProxyEvent): AuthContext {
  const claims = event.requestContext.authorizer?.claims as TokenClaims | undefined;

  if (!claims) {
    throw new UnauthorizedError('No authentication claims found');
  }

  // Check token expiration
  if (claims.exp && claims.exp < Date.now() / 1000) {
    throw new UnauthorizedError('Token has expired');
  }

  // Extract core identifiers
  const userId = claims.sub;
  const tenantId = claims['custom:tenantId'] || claims['custom:tenant_id'];
  const appId = claims['custom:appId'] || claims['custom:app_id'];
  const appUserId = claims['custom:appUserId'] || claims['custom:app_user_id'];
  const email = claims.email || claims['cognito:username'] || '';

  // Validate required claims
  if (!userId) throw new UnauthorizedError('Missing user ID');
  if (!tenantId) throw new UnauthorizedError('Missing tenant ID');

  // App isolation - appId and appUserId required for v4.6.0+
  // Fallback for backward compatibility with pre-v4.6.0 tokens
  const resolvedAppId = appId || extractAppIdFromRoute(event) || 'default';
  const resolvedAppUserId = appUserId || userId; // Fallback to userId for legacy tokens

  // Validate app_id matches route (defense in depth)
  const routeAppId = extractAppIdFromRoute(event);
  if (routeAppId && appId && routeAppId !== appId) {
    throw new ForbiddenError(`Token app_id (${appId}) does not match route (${routeAppId})`)
  }
```

```
  // Extract roles and groups
  const groups = claims['cognito:groups'] || [];
  const roles = claims['custom:role'] ? [claims['custom:role']] : [];

  const isAdmin = groups.some(g =>
    ['super_admin', 'admin', 'operator', 'auditor'].includes(g)
  );
  const isSuperAdmin = groups.includes('super_admin');

  return {
    userId,
    appUserId: resolvedAppUserId,
    tenantId,
    appId: resolvedAppId,
    email,
    roles,
    groups,
    isAdmin,
    isSuperAdmin,
    tokenExpiry: claims.exp || 0,
  };
}

/**
 * Extract app_id from route for validation (v4.6.0)
 */
function extractAppIdFromRoute(event: APIGatewayProxyEvent): string | null {
  // Extract from subdomain: thinktank.domain.com -> thinktank
  const host = event.headers.Host || event.headers['host'];
  if (host) {
    const subdomain = host.split('.')[0];
    if (['thinktank', 'launchboard', 'alwaysme', 'mechanicalmaker'].includes(subdomain)) {
      return subdomain;
    }
  }

  // Extract from path: /api/thinktank/... -> thinktank
  const pathMatch = event.path.match(/^\/api\/(thinktank|launchboard|alwaysme|mechanicalmake
  if (pathMatch) {
    return pathMatch[1];
  }

  return null;
}

/**
```

```typescript
 * Require specific roles
 */
export function requireRoles(auth: AuthContext, requiredRoles: string[]): void {
  const hasRole = requiredRoles.some(role =>
    auth.roles.includes(role) || auth.groups.includes(role)
  );

  if (!hasRole) {
    throw new ForbiddenError(
      `Required roles: ${requiredRoles.join(', ')}`
    );
  }
}

/**
 * Require admin access
 */
export function requireAdmin(auth: AuthContext): void {
  if (!auth.isAdmin) {
    throw new ForbiddenError('Admin access required');
  }
}

/**
 * Require super admin access
 */
export function requireSuperAdmin(auth: AuthContext): void {
  if (!auth.groups.includes('super_admin')) {
    throw new ForbiddenError('Super admin access required');
  }
}

/**
 * Check if user can access tenant
 */
export function canAccessTenant(auth: AuthContext, tenantId: string): boolean {
  // Super admins can access any tenant
  if (auth.groups.includes('super_admin')) {
    return true;
  }

  // Users can only access their own tenant
  return auth.tenantId === tenantId;
}

/**
```

```
 * Require tenant access
 */
export function requireTenantAccess(auth: AuthContext, tenantId: string): void {
  if (!canAccessTenant(auth, tenantId)) {
    throw new ForbiddenError('Access to tenant denied');
  }
}

/**
 * Extract API key from header (for API key auth)
 */
export function extractApiKey(event: APIGatewayProxyEvent): string | undefined {
  return event.headers['X-Api-Key'] || event.headers['x-api-key'];
}

/**
 * Log authentication context (sanitized)
 */
export function logAuthContext(auth: AuthContext, logger: Logger): void {
  logger.info('Authenticated request', {
    userId: auth.userId,
    tenantId: auth.tenantId,
    isAdmin: auth.isAdmin,
    groupCount: auth.groups.length,
  });
}
```

PART 3: DATABASE CLIENT

packages/infrastructure/lambda/shared/db/index.ts

```
export * from './client';
export * from './queries';
export * from './types';
```

packages/infrastructure/lambda/shared/db/types.ts

```
/**
 * Database types matching Aurora PostgreSQL schema
 */

// ============================================================================
// PROVIDERS
// ============================================================================
```

```typescript
export interface DBProvider {
  id: string;
  name: string;
  type: 'external' | 'self-hosted' | 'mid-tier';
  status: 'active' | 'inactive' | 'deprecated';
  hipaa_compliant: boolean;
  baa_available: boolean;
  base_url: string | null;
  auth_type: 'api_key' | 'oauth' | 'iam' | 'none';
  capabilities: string[];
  config: Record<string, unknown>;
  created_at: string;
  updated_at: string;
}

// ============================================================================
// MODELS
// ============================================================================

export interface DBModel {
  id: string;
  provider_id: string;
  name: string;
  display_name: string;
  description: string | null;
  type: 'external' | 'self-hosted';
  specialty: string;
  capabilities: string[];
  context_window: number | null;
  max_output_tokens: number | null;
  supports_functions: boolean;
  supports_vision: boolean;
  supports_streaming: boolean;
  has_thinking_mode: boolean;
  thinking_budget_tokens: number | null;
  pricing: DBModelPricing;
  thermal_state: string | null;
  thermal_config: DBThermalConfig | null;
  status: 'active' | 'inactive' | 'deprecated' | 'coming_soon';
  release_date: string | null;
  deprecation_date: string | null;
  created_at: string;
  updated_at: string;
}

export interface DBModelPricing {
```

```
  input_tokens?: number;
  output_tokens?: number;
  per_image?: number;
  per_minute_audio?: number;
  per_minute_video?: number;
  per_3d_model?: number;
  billed_markup: number;
}

export interface DBThermalConfig {
  state: 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';
  min_instances: number;
  max_instances: number;
  scale_to_zero_after_minutes?: number;
  warmup_time_seconds?: number;
}

// ============================================================================
// TENANTS
// ============================================================================

export interface DBTenant {
  id: string;
  app_id: string;
  name: string;
  domain: string | null;
  status: 'active' | 'suspended' | 'deleted';
  settings: DBTenantSettings;
  phi_config: DBPhiConfig | null;
  created_at: string;
  updated_at: string;
}

export interface DBTenantSettings {
  default_model?: string;
  allowed_providers?: string[];
  max_tokens_per_request?: number;
  rate_limit?: {
    requests_per_minute: number;
    tokens_per_day: number;
  };
}

export interface DBPhiConfig {
  mode: 'auto' | 'manual' | 'disabled';
  categories: Record<string, boolean>;
```

```
  reidentification: {
    allowed: boolean;
    requires_approval: boolean;
    mapping_ttl_hours: number;
  };
}

// ============================================================================
// USAGE
// ============================================================================

export interface DBUsageRecord {
  id: string;
  tenant_id: string;
  user_id: string | null;
  session_id: string | null;
  model_id: string;
  provider_id: string;
  input_tokens: number;
  output_tokens: number;
  total_tokens: number;
  cost: number;
  billed_amount: number;
  request_type: string;
  latency_ms: number;
  created_at: string;
}

// ============================================================================
// AUDIT LOG
// ============================================================================

export interface DBAuditLog {
  id: string;
  tenant_id: string;
  user_id: string | null;
  admin_id: string | null;
  action: string;
  resource_type: string;
  resource_id: string | null;
  details: Record<string, unknown>;
  ip_address: string | null;
  user_agent: string | null;
  created_at: string;
}
```

packages/infrastructure/lambda/shared/db/client.ts

```
/**
 * Aurora PostgreSQL client using Data API
 */

import {
  RDSDataClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
  BeginTransactionCommand,
  CommitTransactionCommand,
  RollbackTransactionCommand,
  Field,
  SqlParameter,
} from '@aws-sdk/client-rds-data';
import {
  SecretsManagerClient,
  GetSecretValueCommand,
} from '@aws-sdk/client-secrets-manager';
import { getConfig } from '../config';
import { Logger } from '../logger';
import { InternalError } from '../errors';

// Initialize clients
const rdsClient = new RDSDataClient({});
const secretsClient = new SecretsManagerClient({});

// Cache database credentials
let dbCredentials: { username: string; password: string } | null = null;

export interface QueryResult<T = Record<string, unknown>> {
  rows: T[];
  rowCount: number;
  columnMetadata?: { name: string; type: string }[];
}

export interface TransactionContext {
  transactionId: string;
}

/**
 * Get database credentials from Secrets Manager
 */
async function getDbCredentials(): Promise<{ username: string; password: string }> {
  if (dbCredentials) return dbCredentials;
```

```
  const config = getConfig();

  try {
    const command = new GetSecretValueCommand({
      SecretId: config.AURORA_SECRET_ARN,
    });

    const response = await secretsClient.send(command);

    if (!response.SecretString) {
      throw new Error('Secret value is empty');
    }

    dbCredentials = JSON.parse(response.SecretString);
    return dbCredentials!;
  } catch (error) {
    throw new InternalError(`Failed to retrieve database credentials: ${error}`);
  }
}

/**
 * Convert JavaScript value to SQL parameter
 */
function toSqlParameter(name: string, value: unknown): SqlParameter {
  if (value === null || value === undefined) {
    return { name, value: { isNull: true } };
  }

  if (typeof value === 'string') {
    return { name, value: { stringValue: value } };
  }

  if (typeof value === 'number') {
    if (Number.isInteger(value)) {
      return { name, value: { longValue: value } };
    }
    return { name, value: { doubleValue: value } };
  }

  if (typeof value === 'boolean') {
    return { name, value: { booleanValue: value } };
  }

  if (Array.isArray(value)) {
    return { name, value: { stringValue: JSON.stringify(value) }, typeHint: 'JSON' };
```

```
  }

  if (typeof value === 'object') {
    return { name, value: { stringValue: JSON.stringify(value) }, typeHint: 'JSON' };
  }

  return { name, value: { stringValue: String(value) } };
}

/**
 * Convert SQL field to JavaScript value
 */
function fromSqlField(field: Field): unknown {
  if (field.isNull) return null;
  if (field.stringValue !== undefined) return field.stringValue;
  if (field.longValue !== undefined) return field.longValue;
  if (field.doubleValue !== undefined) return field.doubleValue;
  if (field.booleanValue !== undefined) return field.booleanValue;
  if (field.blobValue !== undefined) return field.blobValue;
  if (field.arrayValue !== undefined) {
    return field.arrayValue.stringValues ||
           field.arrayValue.longValues ||
           field.arrayValue.doubleValues ||
           field.arrayValue.booleanValues ||
           [];
  }
  return null;
}

/**
 * Execute a SQL query
 */
export async function query<T = Record<string, unknown>>(
  sql: string,
  params: Record<string, unknown> = {},
  logger?: Logger,
  transactionId?: string
): Promise<QueryResult<T>> {
  const config = getConfig();
  const startTime = Date.now();

  try {
    const command = new ExecuteStatementCommand({
      resourceArn: config.AURORA_CLUSTER_ARN,
      secretArn: config.AURORA_SECRET_ARN,
      database: 'radiant',
```

```
      sql,
      parameters: Object.entries(params).map(([name, value]) =>
        toSqlParameter(name, value)
      ),
      includeResultMetadata: true,
      transactionId,
    });

    const response = await rdsClient.send(command);
    const duration = Date.now() - startTime;

    if (logger) {
      logger.debug('Database query executed', {
        sql: sql.substring(0, 100),
        duration,
        rowCount: response.numberOfRecordsUpdated,
      });
    }

    // Convert response to rows
    const rows: T[] = [];

    if (response.records && response.columnMetadata) {
      for (const record of response.records) {
        const row: Record<string, unknown> = {};

        for (let i = 0; i < response.columnMetadata.length; i++) {
          const columnName = response.columnMetadata[i].name || `col${i}`;
          row[columnName] = fromSqlField(record[i]);
        }

        rows.push(row as T);
      }
    }

    return {
      rows,
      rowCount: response.numberOfRecordsUpdated || rows.length,
      columnMetadata: response.columnMetadata?.map(col => ({
        name: col.name || '',
        type: col.typeName || '',
      })),
    };
  } catch (error) {
    const duration = Date.now() - startTime;
```

```
    if (logger) {
      logger.error('Database query failed', error as Error, {
        sql: sql.substring(0, 100),
        duration,
      });
    }

    throw new InternalError(`Database query failed: ${error}`);
  }
}

/**
 * Execute a single row query
 */
export async function queryOne<T = Record<string, unknown>>(
  sql: string,
  params: Record<string, unknown> = {},
  logger?: Logger
): Promise<T | null> {
  const result = await query<T>(sql, params, logger);
  return result.rows[0] || null;
}

/**
 * Begin a transaction
 */
export async function beginTransaction(logger?: Logger): Promise<TransactionContext> {
  const config = getConfig();

  try {
    const command = new BeginTransactionCommand({
      resourceArn: config.AURORA_CLUSTER_ARN,
      secretArn: config.AURORA_SECRET_ARN,
      database: 'radiant',
    });

    const response = await rdsClient.send(command);

    if (!response.transactionId) {
      throw new Error('No transaction ID returned');
    }

    if (logger) {
      logger.debug('Transaction started', { transactionId: response.transactionId });
    }
```

```
    return { transactionId: response.transactionId };
  } catch (error) {
    throw new InternalError(`Failed to begin transaction: ${error}`);
  }
}

/**
 * Commit a transaction
 */
export async function commitTransaction(
  ctx: TransactionContext,
  logger?: Logger
): Promise<void> {
  const config = getConfig();

  try {
    const command = new CommitTransactionCommand({
      resourceArn: config.AURORA_CLUSTER_ARN,
      secretArn: config.AURORA_SECRET_ARN,
      transactionId: ctx.transactionId,
    });

    await rdsClient.send(command);

    if (logger) {
      logger.debug('Transaction committed', { transactionId: ctx.transactionId });
    }
  } catch (error) {
    throw new InternalError(`Failed to commit transaction: ${error}`);
  }
}

/**
 * Rollback a transaction
 */
export async function rollbackTransaction(
  ctx: TransactionContext,
  logger?: Logger
): Promise<void> {
  const config = getConfig();

  try {
    const command = new RollbackTransactionCommand({
      resourceArn: config.AURORA_CLUSTER_ARN,
      secretArn: config.AURORA_SECRET_ARN,
      transactionId: ctx.transactionId,
```

```
    });

    await rdsClient.send(command);

    if (logger) {
      logger.debug('Transaction rolled back', { transactionId: ctx.transactionId });
    }
  } catch (error) {
    // Log but don't throw - rollback errors are usually not critical
    if (logger) {
      logger.warn('Transaction rollback failed', {
        transactionId: ctx.transactionId,
        error: String(error),
      });
    }
  }
}

/**
 * Execute a callback within a transaction
 */
export async function withTransaction<T>(
  callback: (transactionId: string) => Promise<T>,
  logger?: Logger
): Promise<T> {
  const ctx = await beginTransaction(logger);

  try {
    const result = await callback(ctx.transactionId);
    await commitTransaction(ctx, logger);
    return result;
  } catch (error) {
    await rollbackTransaction(ctx, logger);
    throw error;
  }
}
```

packages/infrastructure/lambda/shared/db/queries.ts

```
/**
 * Pre-built database queries
 */

import { query, queryOne, QueryResult } from './client';
import { DBProvider, DBModel, DBTenant, DBUsageRecord, DBAuditLog } from './types';
import { Logger } from '../logger';
```

```typescript
import { NotFoundError } from '../errors';

// ============================================================================
// PROVIDERS
// ============================================================================

export async function getProviders(
  filters: {
    type?: string;
    status?: string;
    hipaaCompliant?: boolean;
    limit?: number;
    offset?: number;
  } = {},
  logger?: Logger
): Promise<QueryResult<DBProvider>> {
  let sql = `
    SELECT * FROM providers
    WHERE 1=1
  `;
  const params: Record<string, unknown> = {};

  if (filters.type) {
    sql += ` AND type = :type`;
    params.type = filters.type;
  }

  if (filters.status) {
    sql += ` AND status = :status`;
    params.status = filters.status;
  }

  if (filters.hipaaCompliant !== undefined) {
    sql += ` AND hipaa_compliant = :hipaaCompliant`;
    params.hipaaCompliant = filters.hipaaCompliant;
  }

  sql += ` ORDER BY name ASC`;

  if (filters.limit) {
    sql += ` LIMIT :limit`;
    params.limit = filters.limit;
  }

  if (filters.offset) {
    sql += ` OFFSET :offset`;
```

```
    params.offset = filters.offset;
  }

  return query<DBProvider>(sql, params, logger);
}

export async function getProviderById(
  id: string,
  logger?: Logger
): Promise<DBProvider> {
  const result = await queryOne<DBProvider>(
    `SELECT * FROM providers WHERE id = :id`,
    { id },
    logger
  );

  if (!result) {
    throw new NotFoundError(`Provider ${id}`);
  }

  return result;
}

export async function updateProvider(
  id: string,
  updates: Partial<Pick<DBProvider, 'status' | 'hipaa_compliant' | 'config'>>,
  logger?: Logger
): Promise<DBProvider> {
  const setClauses: string[] = ['updated_at = NOW()'];
  const params: Record<string, unknown> = { id };

  if (updates.status !== undefined) {
    setClauses.push('status = :status');
    params.status = updates.status;
  }

  if (updates.hipaa_compliant !== undefined) {
    setClauses.push('hipaa_compliant = :hipaaCompliant');
    params.hipaaCompliant = updates.hipaa_compliant;
  }

  if (updates.config !== undefined) {
    setClauses.push('config = :config');
    params.config = updates.config;
  }
```

```
  const sql = `
    UPDATE providers
    SET ${setClauses.join(', ')}
    WHERE id = :id
    RETURNING *
  `;

  const result = await queryOne<DBProvider>(sql, params, logger);

  if (!result) {
    throw new NotFoundError(`Provider ${id}`);
  }

  return result;
}

// ============================================================================
// MODELS
// ============================================================================

export async function getModels(
  filters: {
    providerId?: string;
    specialty?: string;
    status?: string;
    type?: string;
    supportsVision?: boolean;
    supportsStreaming?: boolean;
    limit?: number;
    offset?: number;
  } = {},
  logger?: Logger
): Promise<QueryResult<DBModel>> {
  let sql = `
    SELECT * FROM models
    WHERE 1=1
  `;
  const params: Record<string, unknown> = {};

  if (filters.providerId) {
    sql += ` AND provider_id = :providerId`;
    params.providerId = filters.providerId;
  }

  if (filters.specialty) {
    sql += ` AND specialty = :specialty`;
```

```
    params.specialty = filters.specialty;
  }

  if (filters.status) {
    sql += ` AND status = :status`;
    params.status = filters.status;
  }

  if (filters.type) {
    sql += ` AND type = :type`;
    params.type = filters.type;
  }

  if (filters.supportsVision !== undefined) {
    sql += ` AND supports_vision = :supportsVision`;
    params.supportsVision = filters.supportsVision;
  }

  if (filters.supportsStreaming !== undefined) {
    sql += ` AND supports_streaming = :supportsStreaming`;
    params.supportsStreaming = filters.supportsStreaming;
  }

  sql += ` ORDER BY display_name ASC`;

  if (filters.limit) {
    sql += ` LIMIT :limit`;
    params.limit = filters.limit;
  }

  if (filters.offset) {
    sql += ` OFFSET :offset`;
    params.offset = filters.offset;
  }

  return query<DBModel>(sql, params, logger);
}

export async function getModelById(
  id: string,
  logger?: Logger
): Promise<DBModel> {
  const result = await queryOne<DBModel>(
    `SELECT * FROM models WHERE id = :id`,
    { id },
    logger
```

```
  );

  if (!result) {
    throw new NotFoundError(`Model ${id}`);
  }

  return result;
}

export async function getModelByName(
  name: string,
  logger?: Logger
): Promise<DBModel | null> {
  return queryOne<DBModel>(
    `SELECT * FROM models WHERE name = :name AND status = 'active'`,
    { name },
    logger
  );
}

export async function updateModel(
  id: string,
  updates: Partial<Pick<DBModel, 'status' | 'thermal_state' | 'thermal_config' | 'display_na
  logger?: Logger
): Promise<DBModel> {
  const setClauses: string[] = ['updated_at = NOW()'];
  const params: Record<string, unknown> = { id };

  if (updates.status !== undefined) {
    setClauses.push('status = :status');
    params.status = updates.status;
  }

  if (updates.thermal_state !== undefined) {
    setClauses.push('thermal_state = :thermalState');
    params.thermalState = updates.thermal_state;
  }

  if (updates.thermal_config !== undefined) {
    setClauses.push('thermal_config = :thermalConfig');
    params.thermalConfig = updates.thermal_config;
  }

  if (updates.display_name !== undefined) {
    setClauses.push('display_name = :displayName');
    params.displayName = updates.display_name;
```

```
  }

  if (updates.description !== undefined) {
    setClauses.push('description = :description');
    params.description = updates.description;
  }

  const sql = `
    UPDATE models
    SET ${setClauses.join(', ')}
    WHERE id = :id
    RETURNING *
  `;

  const result = await queryOne<DBModel>(sql, params, logger);

  if (!result) {
    throw new NotFoundError(`Model ${id}`);
  }

  return result;
}

// ============================================================================
// TENANTS
// ============================================================================

export async function getTenantById(
  id: string,
  logger?: Logger
): Promise<DBTenant> {
  const result = await queryOne<DBTenant>(
    `SELECT * FROM tenants WHERE id = :id AND status = 'active'`,
    { id },
    logger
  );

  if (!result) {
    throw new NotFoundError(`Tenant ${id}`);
  }

  return result;
}

export async function getTenantPhiConfig(
  tenantId: string,
```

```
  logger?: Logger
): Promise<DBTenant['phi_config']> {
  const tenant = await getTenantById(tenantId, logger);
  return tenant.phi_config;
}

// ============================================================================
// USAGE
// ============================================================================

export async function recordUsage(
  usage: Omit<DBUsageRecord, 'id' | 'created_at'>,
  logger?: Logger
): Promise<DBUsageRecord> {
  const sql = `
    INSERT INTO usage_records (
      id, tenant_id, user_id, session_id, model_id, provider_id,
      input_tokens, output_tokens, total_tokens, cost, billed_amount,
      request_type, latency_ms
    ) VALUES (
      gen_random_uuid(), :tenantId, :userId, :sessionId, :modelId, :providerId,
      :inputTokens, :outputTokens, :totalTokens, :cost, :billedAmount,
      :requestType, :latencyMs
    )
    RETURNING *
  `;

  const result = await queryOne<DBUsageRecord>(sql, {
    tenantId: usage.tenant_id,
    userId: usage.user_id,
    sessionId: usage.session_id,
    modelId: usage.model_id,
    providerId: usage.provider_id,
    inputTokens: usage.input_tokens,
    outputTokens: usage.output_tokens,
    totalTokens: usage.total_tokens,
    cost: usage.cost,
    billedAmount: usage.billed_amount,
    requestType: usage.request_type,
    latencyMs: usage.latency_ms,
  }, logger);

  return result!;
}

// ============================================================================
```

```typescript
// AUDIT LOG
// ============================================================================

export async function createAuditLog(
  log: Omit<DBAuditLog, 'id' | 'created_at'>,
  logger?: Logger
): Promise<DBAuditLog> {
  const sql = `
    INSERT INTO audit_logs (
      id, tenant_id, user_id, admin_id, action, resource_type,
      resource_id, details, ip_address, user_agent
    ) VALUES (
      gen_random_uuid(), :tenantId, :userId, :adminId, :action, :resourceType,
      :resourceId, :details, :ipAddress, :userAgent
    )
    RETURNING *
  `;

  const result = await queryOne<DBAuditLog>(sql, {
    tenantId: log.tenant_id,
    userId: log.user_id,
    adminId: log.admin_id,
    action: log.action,
    resourceType: log.resource_type,
    resourceId: log.resource_id,
    details: log.details,
    ipAddress: log.ip_address,
    userAgent: log.user_agent,
  }, logger);

  return result!;
}
```

PART 4: LITELLM CLIENT

packages/infrastructure/lambda/shared/litellm/index.ts

```typescript
export * from './client';
export * from './types';
```

packages/infrastructure/lambda/shared/litellm/types.ts

```typescript
/**
 * LiteLLM API types
 */
```

```typescript
// ============================================================================
// CHAT COMPLETION
// ============================================================================

export interface ChatCompletionRequest {
  model: string;
  messages: ChatMessage[];
  max_tokens?: number;
  temperature?: number;
  top_p?: number;
  stream?: boolean;
  stop?: string | string[];
  presence_penalty?: number;
  frequency_penalty?: number;
  user?: string;
  metadata?: Record<string, unknown>;
}

export interface ChatMessage {
  role: 'system' | 'user' | 'assistant' | 'function' | 'tool';
  content: string | ContentPart[];
  name?: string;
  function_call?: FunctionCall;
  tool_calls?: ToolCall[];
}

export interface ContentPart {
  type: 'text' | 'image_url';
  text?: string;
  image_url?: {
    url: string;
    detail?: 'low' | 'high' | 'auto';
  };
}

export interface FunctionCall {
  name: string;
  arguments: string;
}

export interface ToolCall {
  id: string;
  type: 'function';
  function: FunctionCall;
}
```

```typescript
export interface ChatCompletionResponse {
  id: string;
  object: 'chat.completion';
  created: number;
  model: string;
  choices: ChatChoice[];
  usage: TokenUsage;
  system_fingerprint?: string;
}

export interface ChatChoice {
  index: number;
  message: ChatMessage;
  finish_reason: 'stop' | 'length' | 'function_call' | 'tool_calls' | 'content_filter' | nul
  logprobs?: unknown;
}

export interface TokenUsage {
  prompt_tokens: number;
  completion_tokens: number;
  total_tokens: number;
}

// ============================================================================
// STREAMING
// ============================================================================

export interface ChatCompletionChunk {
  id: string;
  object: 'chat.completion.chunk';
  created: number;
  model: string;
  choices: StreamChoice[];
  usage?: TokenUsage;
}

export interface StreamChoice {
  index: number;
  delta: {
    role?: string;
    content?: string;
    function_call?: Partial<FunctionCall>;
    tool_calls?: Partial<ToolCall>[];
  };
  finish_reason: string | null;
```

```typescript
}

// ============================================================================
// EMBEDDINGS
// ============================================================================

export interface EmbeddingRequest {
  model: string;
  input: string | string[];
  encoding_format?: 'float' | 'base64';
  dimensions?: number;
  user?: string;
}

export interface EmbeddingResponse {
  object: 'list';
  data: EmbeddingData[];
  model: string;
  usage: {
    prompt_tokens: number;
    total_tokens: number;
  };
}

export interface EmbeddingData {
  object: 'embedding';
  index: number;
  embedding: number[];
}

// ============================================================================
// MODELS
// ============================================================================

export interface LiteLLMModel {
  id: string;
  object: 'model';
  created: number;
  owned_by: string;
}

export interface LiteLLMModelList {
  object: 'list';
  data: LiteLLMModel[];
}
```

```typescript
// ============================================================================
// HEALTH
// ============================================================================

export interface HealthResponse {
  status: 'healthy' | 'unhealthy';
  version?: string;
  models?: string[];
}

// ============================================================================
// ERRORS
// ============================================================================

export interface LiteLLMError {
  error: {
    message: string;
    type: string;
    param?: string;
    code?: string;
  };
}
```

packages/infrastructure/lambda/shared/litellm/client.ts

```typescript
/**
 * LiteLLM HTTP client
 */

import { getConfig } from '../config';
import { Logger } from '../logger';
import { ProviderError, ServiceUnavailableError, RateLimitError } from '../errors';
import {
  ChatCompletionRequest,
  ChatCompletionResponse,
  ChatCompletionChunk,
  EmbeddingRequest,
  EmbeddingResponse,
  LiteLLMModelList,
  HealthResponse,
  LiteLLMError,
} from './types';

const DEFAULT_TIMEOUT = 120000; // 2 minutes for AI requests

interface FetchOptions {
```

```
  method?: string;
  body?: unknown;
  timeout?: number;
  stream?: boolean;
}

/**
 * Make HTTP request to LiteLLM
 */
async function fetchLiteLLM<T>(
  path: string,
  options: FetchOptions = {},
  logger?: Logger
): Promise<T> {
  const config = getConfig();
  const url = `${config.LITELLM_URL}${path}`;
  const timeout = options.timeout || DEFAULT_TIMEOUT;

  const controller = new AbortController();
  const timeoutId = setTimeout(() => controller.abort(), timeout);

  try {
    const startTime = Date.now();

    const response = await fetch(url, {
      method: options.method || 'GET',
      headers: {
        'Content-Type': 'application/json',
        Accept: options.stream ? 'text/event-stream' : 'application/json',
      },
      body: options.body ? JSON.stringify(options.body) : undefined,
      signal: controller.signal,
    });

    const duration = Date.now() - startTime;

    if (logger) {
      logger.debug('LiteLLM request completed', {
        path,
        status: response.status,
        duration,
      });
    }

    if (!response.ok) {
      const errorBody = await response.text();
```

```
        let errorMessage = `LiteLLM error: ${response.status}`;

        try {
          const errorJson = JSON.parse(errorBody) as LiteLLMError;
          errorMessage = errorJson.error?.message || errorMessage;
        } catch {
          errorMessage = errorBody || errorMessage;
        }

        if (response.status === 429) {
          const retryAfter = parseInt(response.headers.get('Retry-After') || '60');
          throw new RateLimitError(retryAfter);
        }

        if (response.status >= 500) {
          throw new ServiceUnavailableError(errorMessage);
        }

        throw new ProviderError(errorMessage, 'litellm');
      }

      if (options.stream) {
        return response as unknown as T;
      }

      return await response.json() as T;
    } catch (error) {
      if (error instanceof Error && error.name === 'AbortError') {
        throw new ServiceUnavailableError('LiteLLM request timed out');
      }

      if (error instanceof ProviderError ||
          error instanceof ServiceUnavailableError ||
          error instanceof RateLimitError) {
        throw error;
      }

      throw new ServiceUnavailableError(`LiteLLM connection failed: ${error}`);
    } finally {
      clearTimeout(timeoutId);
    }
  }

  /**
   * Create chat completion
   */
```

```
export async function createChatCompletion(
  request: ChatCompletionRequest,
  logger?: Logger
): Promise<ChatCompletionResponse> {
  return fetchLiteLLM<ChatCompletionResponse>(
    '/v1/chat/completions',
    {
      method: 'POST',
      body: request,
    },
    logger
  );
}

/**
 * Create streaming chat completion
 */
export async function* streamChatCompletion(
  request: ChatCompletionRequest,
  logger?: Logger
): AsyncGenerator<ChatCompletionChunk> {
  const streamRequest = { ...request, stream: true };

  const response = await fetchLiteLLM<Response>(
    '/v1/chat/completions',
    {
      method: 'POST',
      body: streamRequest,
      stream: true,
    },
    logger
  );

  const reader = response.body?.getReader();
  if (!reader) {
    throw new ServiceUnavailableError('No response body for streaming');
  }

  const decoder = new TextDecoder();
  let buffer = '';

  try {
    while (true) {
      const { done, value } = await reader.read();

      if (done) break;
```

```
      buffer += decoder.decode(value, { stream: true });
      const lines = buffer.split('\n');
      buffer = lines.pop() || '';

      for (const line of lines) {
        const trimmed = line.trim();

        if (!trimmed || !trimmed.startsWith('data: ')) continue;

        const data = trimmed.slice(6);

        if (data === '[DONE]') {
          return;
        }

        try {
          const chunk = JSON.parse(data) as ChatCompletionChunk;
          yield chunk;
        } catch {
          if (logger) {
            logger.warn('Failed to parse SSE chunk', { data });
          }
        }
      }
    }
  } finally {
    reader.releaseLock();
  }
}

/**
 * Create embeddings
 */
export async function createEmbedding(
  request: EmbeddingRequest,
  logger?: Logger
): Promise<EmbeddingResponse> {
  return fetchLiteLLM<EmbeddingResponse>(
    '/v1/embeddings',
    {
      method: 'POST',
      body: request,
    },
    logger
  );
```

```
}

/**
 * List available models
 */
export async function listModels(logger?: Logger): Promise<LiteLLMModelList> {
  return fetchLiteLLM<LiteLLMModelList>('/v1/models', {}, logger);
}

/**
 * Health check
 */
export async function checkHealth(logger?: Logger): Promise<HealthResponse> {
  try {
    const response = await fetchLiteLLM<HealthResponse>(
      '/health',
      { timeout: 5000 },
      logger
    );
    return response;
  } catch {
    return { status: 'unhealthy' };
  }
}

/**
 * Calculate cost for a completion
 */
export function calculateCost(
  usage: { prompt_tokens: number; completion_tokens: number },
  pricing: { input_tokens?: number; output_tokens?: number; billed_markup: number }
): { cost: number; billed: number } {
  const inputCost = (usage.prompt_tokens / 1_000_000) * (pricing.input_tokens || 0);
  const outputCost = (usage.completion_tokens / 1_000_000) * (pricing.output_tokens || 0);
  const cost = inputCost + outputCost;
  const billed = cost * (1 + pricing.billed_markup);

  return { cost, billed };
}
```

PART 5: PHI SANITIZATION

packages/infrastructure/lambda/shared/phi/index.ts

```typescript
export * from './sanitizer';
export * from './patterns';
export * from './types';
```

packages/infrastructure/lambda/shared/phi/types.ts

```typescript
/**
 * PHI (Protected Health Information) types
 */

export type PHICategory =
  | 'NAME'
  | 'SSN'
  | 'DOB'
  | 'ADDRESS'
  | 'PHONE'
  | 'EMAIL'
  | 'DIAGNOSIS'
  | 'TREATMENT'
  | 'MEDICAL_RECORD'
  | 'INSURANCE_ID';

export interface PHIConfig {
  mode: 'auto' | 'manual' | 'disabled';
  categories: Record<PHICategory, boolean>;
  reidentification: {
    allowed: boolean;
    requires_approval: boolean;
    mapping_ttl_hours: number;
  };
}

export interface PHIMatch {
  category: PHICategory;
  original: string;
  placeholder: string;
  startIndex: number;
  endIndex: number;
  confidence: number;
}

export interface SanitizationResult {
  sanitizedText: string;
```

```
  matches: PHIMatch[];
  mappingId: string;
}

export interface ReidentificationResult {
  originalText: string;
  matches: PHIMatch[];
}

export interface PHIMapping {
  id: string;
  tenant_id: string;
  session_id: string | null;
  mappings: Record<string, string>; // placeholder -> original
  created_at: string;
  expires_at: string;
}

export const DEFAULT_PHI_CONFIG: PHIConfig = {
  mode: 'auto',
  categories: {
    NAME: true,
    SSN: true,
    DOB: true,
    ADDRESS: true,
    PHONE: true,
    EMAIL: true,
    DIAGNOSIS: false, // Often needed for AI analysis
    TREATMENT: false, // Often needed for AI analysis
    MEDICAL_RECORD: true,
    INSURANCE_ID: true,
  },
  reidentification: {
    allowed: true,
    requires_approval: true,
    mapping_ttl_hours: 24,
  },
};
```

packages/infrastructure/lambda/shared/phi/patterns.ts

```
/**
 * PHI detection patterns
 */

import { PHICategory } from './types';
```

```typescript
export interface PHIPattern {
  category: PHICategory;
  patterns: RegExp[];
  validator?: (match: string) => boolean;
  confidence: number;
}

/**
 * PHI detection patterns by category
 */
export const PHI_PATTERNS: PHIPattern[] = [
  // Social Security Numbers
  {
    category: 'SSN',
    patterns: [
      /\b\d{3}-\d{2}-\d{4}\b/g,
      /\b\d{3}\s\d{2}\s\d{4}\b/g,
      /\bSSN[:\s]*\d{3}[-\s]?\d{2}[-\s]?\d{4}\b/gi,
    ],
    validator: (match) => {
      const digits = match.replace(/\D/g, '');
      if (digits.length !== 9) return false;
      // Invalid SSN patterns
      if (digits.startsWith('000') || digits.startsWith('666')) return false;
      if (digits.substring(0, 3) === '900' && parseInt(digits.substring(0, 3)) <= 999) retu
      return true;
    },
    confidence: 0.95,
  },

  // Dates of Birth
  {
    category: 'DOB',
    patterns: [
      /\b(?:DOB|Date of Birth|Birthday|Born)[:\s]*(\d{1,2}[-\/]\d{1,2}[-\/]\d{2,4})\b/gi,
      /\b(?:DOB|Date of Birth|Birthday|Born)[:\s]*([A-Z][a-z]+\s+\d{1,2},?\s+\d{4})\b/gi,
    ],
    confidence: 0.90,
  },

  // Phone Numbers
  {
    category: 'PHONE',
    patterns: [
      /\b\(?[2-9]\d{2}\)?[-.\s]?\d{3}[-.\s]?\d{4}\b/g,
```

```
    /\b(?:Phone|Tel|Mobile|Cell)[:\s]*\(?[2-9]\d{2}\)?[-.\s]?\d{3}[-.\s]?\d{4}\b/gi,
    /\b\+1[-.\s]?\(?[2-9]\d{2}\)?[-.\s]?\d{3}[-.\s]?\d{4}\b/g,
  ],
  confidence: 0.85,
},

// Email Addresses
{
  category: 'EMAIL',
  patterns: [
    /\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b/g,
  ],
  validator: (match) => {
    // Exclude common system emails
    const systemDomains = ['YOUR_DOMAIN.com', 'test.com', 'localhost'];
    const domain = match.split('@')[1]?.toLowerCase();
    return !systemDomains.includes(domain);
  },
  confidence: 0.90,
},

// Medical Record Numbers
{
  category: 'MEDICAL_RECORD',
  patterns: [
    /\b(?:MRN|Medical Record|Patient ID)[:\s#]*([A-Z0-9]{6,12})\b/gi,
    /\bMRN[:\s#]*\d{6,12}\b/gi,
  ],
  confidence: 0.85,
},

// Insurance IDs
{
  category: 'INSURANCE_ID',
  patterns: [
    /\b(?:Insurance ID|Policy Number|Member ID)[:\s#]*([A-Z0-9]{8,15})\b/gi,
    /\b(?:Group|Plan)\s*(?:Number|ID|#)[:\s]*([A-Z0-9]{6,12})\b/gi,
  ],
  confidence: 0.80,
},

// Street Addresses
{
  category: 'ADDRESS',
  patterns: [
    /\b\d{1,5}\s+[A-Za-z]+(?:\s+[A-Za-z]+)*\s+(?:Street|St|Avenue|Ave|Road|Rd|Boulevard|Bl
```

```
      /\b(?:P\.?O\.?\s*Box|PO Box)\s+\d+\b/gi,
    ],
    confidence: 0.75,
  },

  // Names (context-dependent)
  {
    category: 'NAME',
    patterns: [
      /\b(?:Patient|Client|Name)[:\s]*([A-Z][a-z]+(?:\s+[A-Z][a-z]+)+)\b/gi,
      /\b(?:Dr\.|Doctor|Mr\.|Mrs\.|Ms\.)\s+([A-Z][a-z]+(?:\s+[A-Z][a-z]+)*)\b/g,
    ],
    confidence: 0.70,
  },

  // Diagnoses (ICD codes and common conditions)
  {
    category: 'DIAGNOSIS',
    patterns: [
      /\b(?:ICD-?10)[:\s]*([A-Z]\d{2}(?:\.\d{1,4})?)\b/gi,
      /\b(?:Diagnosis|Dx)[:\s]*([A-Za-z\s]+(?:syndrome|disease|disorder|condition))\b/gi,
    ],
    confidence: 0.80,
  },

  // Treatments
  {
    category: 'TREATMENT',
    patterns: [
      /\b(?:Rx|Prescription|Medication)[:\s]*([A-Za-z]+(?:\s+\d+\s*mg)?)\b/gi,
      /\b(?:Treatment|Procedure)[:\s]*([A-Za-z\s]+(?:ectomy|plasty|scopy|therapy))\b/gi,
    ],
    confidence: 0.75,
  },
];

/**
 * Get patterns for enabled categories
 */
export function getEnabledPatterns(
  categories: Record<PHICategory, boolean>
): PHIPattern[] {
  return PHI_PATTERNS.filter(pattern => categories[pattern.category]);
}
```

packages/infrastructure/lambda/shared/phi/sanitizer.ts

```
/**
 * PHI sanitization engine
 */

import { v4 as uuid } from 'uuid';
import {
  PHIConfig,
  PHICategory,
  PHIMatch,
  SanitizationResult,
  ReidentificationResult,
  PHIMapping,
  DEFAULT_PHI_CONFIG,
} from './types';
import { getEnabledPatterns, PHIPattern } from './patterns';
import { Logger } from '../logger';

// Placeholder format: [PHI_CATEGORY_INDEX]
const PLACEHOLDER_PREFIX = '[PHI_';
const PLACEHOLDER_SUFFIX = ']';

/**
 * Generate a placeholder for a PHI match
 */
function generatePlaceholder(category: PHICategory, index: number): string {
  return `${PLACEHOLDER_PREFIX}${category}_${index}${PLACEHOLDER_SUFFIX}`;
}

/**
 * Parse a placeholder back to its parts
 */
function parsePlaceholder(placeholder: string): { category: PHICategory; index: number } | n
  const match = placeholder.match(/\[PHI_([A-Z_]+)_(\d+)\]/);
  if (!match) return null;
  return {
    category: match[1] as PHICategory,
    index: parseInt(match[2]),
  };
}

/**
 * Detect PHI in text
 */
export function detectPHI(
```

```
  text: string,
  config: PHIConfig = DEFAULT_PHI_CONFIG,
  logger?: Logger
): PHIMatch[] {
  if (config.mode === 'disabled') {
    return [];
  }

  const matches: PHIMatch[] = [];
  const patterns = getEnabledPatterns(config.categories);
  const categoryCounters: Record<string, number> = {};

  for (const patternDef of patterns) {
    for (const regex of patternDef.patterns) {
      // Reset regex lastIndex
      regex.lastIndex = 0;
      let match: RegExpExecArray | null;

      while ((match = regex.exec(text)) !== null) {
        const original = match[1] || match[0];

        // Skip if validator fails
        if (patternDef.validator && !patternDef.validator(original)) {
          continue;
        }

        // Check for overlapping matches
        const startIndex = text.indexOf(original, match.index);
        const endIndex = startIndex + original.length;

        const isOverlapping = matches.some(
          m => (startIndex >= m.startIndex && startIndex < m.endIndex) ||
               (endIndex > m.startIndex && endIndex <= m.endIndex)
        );

        if (isOverlapping) continue;

        // Generate unique placeholder
        categoryCounters[patternDef.category] = (categoryCounters[patternDef.category] || 0)
        const placeholder = generatePlaceholder(
          patternDef.category,
          categoryCounters[patternDef.category]
        );

        matches.push({
          category: patternDef.category,
```

```
          original,
          placeholder,
          startIndex,
          endIndex,
          confidence: patternDef.confidence,
        });
      }
    }
  }

  // Sort by start index (descending) for safe replacement
  matches.sort((a, b) => b.startIndex - a.startIndex);

  if (logger && matches.length > 0) {
    logger.info('PHI detected', {
      matchCount: matches.length,
      categories: [...new Set(matches.map(m => m.category))],
    });
  }

  return matches;
}

/**
 * Sanitize PHI in text
 */
export function sanitizePHI(
  text: string,
  config: PHIConfig = DEFAULT_PHI_CONFIG,
  logger?: Logger
): SanitizationResult {
  const matches = detectPHI(text, config, logger);

  if (matches.length === 0) {
    return {
      sanitizedText: text,
      matches: [],
      mappingId: '',
    };
  }

  let sanitizedText = text;

  // Replace in reverse order to preserve indices
  for (const match of matches) {
    sanitizedText =
```

```
      sanitizedText.substring(0, match.startIndex) +
      match.placeholder +
      sanitizedText.substring(match.endIndex);
  }

  // Generate mapping ID for re-identification
  const mappingId = uuid();

  return {
    sanitizedText,
    matches: matches.reverse(), // Return in original order
    mappingId,
  };
}

/**
 * Re-identify PHI in text
 */
export function reidentifyPHI(
  sanitizedText: string,
  mapping: PHIMapping,
  logger?: Logger
): ReidentificationResult {
  let originalText = sanitizedText;
  const matches: PHIMatch[] = [];

  // Find all placeholders in the text
  const placeholderRegex = /\[PHI_[A-Z_]+_\d+\]/g;
  let match: RegExpExecArray | null;

  while ((match = placeholderRegex.exec(sanitizedText)) !== null) {
    const placeholder = match[0];
    const original = mapping.mappings[placeholder];

    if (original) {
      const parsed = parsePlaceholder(placeholder);

      if (parsed) {
        matches.push({
          category: parsed.category,
          original,
          placeholder,
          startIndex: match.index,
          endIndex: match.index + placeholder.length,
          confidence: 1.0,
        });
```

```
      }
    }
  }

  // Replace in reverse order
  matches.sort((a, b) => b.startIndex - a.startIndex);

  for (const m of matches) {
    originalText =
      originalText.substring(0, m.startIndex) +
      m.original +
      originalText.substring(m.endIndex);
  }

  if (logger) {
    logger.info('PHI re-identified', {
      matchCount: matches.length,
    });
  }

  return {
    originalText,
    matches: matches.reverse(),
  };
}

/**
 * Create a mapping record for storage
 */
export function createMappingRecord(
  tenantId: string,
  sessionId: string | null,
  result: SanitizationResult,
  ttlHours: number
): PHIMapping {
  const mappings: Record<string, string> = {};

  for (const match of result.matches) {
    mappings[match.placeholder] = match.original;
  }

  const now = new Date();
  const expiresAt = new Date(now.getTime() + ttlHours * 60 * 60 * 1000);

  return {
    id: result.mappingId,
```

```
    tenant_id: tenantId,
    session_id: sessionId,
    mappings,
    created_at: now.toISOString(),
    expires_at: expiresAt.toISOString(),
  };
}

/**
 * Check if text contains any PHI
 */
export function containsPHI(
  text: string,
  config: PHIConfig = DEFAULT_PHI_CONFIG
): boolean {
  const matches = detectPHI(text, config);
  return matches.length > 0;
}

/**
 * Get PHI summary for audit logging
 */
export function getPHISummary(
  matches: PHIMatch[]
): Record<PHICategory, number> {
  const summary: Record<string, number> = {};

  for (const match of matches) {
    summary[match.category] = (summary[match.category] || 0) + 1;
  }

  return summary as Record<PHICategory, number>;
}
```

PART 6: ROUTER LAMBDA

packages/infrastructure/lambda/api/router.ts

```
/**
 * Router Lambda - Main API entry point
 *
 * Handles:
 * - Health checks
 * - Request routing
 * - CORS preflight
```

```
 * - Error handling
 */

import type {
  APIGatewayProxyEvent,
  APIGatewayProxyResult,
  Context,
} from 'aws-lambda';
import { Logger } from '../shared/logger';
import { getConfig } from '../shared/config';
import { success, handleError } from '../shared/response';
import { NotFoundError } from '../shared/errors';
import { checkHealth as checkLiteLLMHealth } from '../shared/litellm';
import { query } from '../shared/db';

// Initialize logger
const logger = new Logger({ handler: 'router' });

/**
 * Main handler
 */
export async function handler(
  event: APIGatewayProxyEvent,
  context: Context
): Promise<APIGatewayProxyResult> {
  // Set request context
  const requestLogger = logger.child({
    requestId: context.awsRequestId,
    path: event.path,
    method: event.httpMethod,
  });

  try {
    const config = getConfig();

    requestLogger.info('Request received', {
      queryParams: event.queryStringParameters,
      hasBody: !!event.body,
    });

    // Route based on path
    const path = event.path.replace(/^\/api\/v2/, '');

    switch (true) {
      // Health check
      case path === '/health' || path === '/':
```

```
        return await handleHealthCheck(requestLogger);

      // Ready check (deep health)
      case path === '/ready':
        return await handleReadyCheck(requestLogger);

      // Version info
      case path === '/version':
        return handleVersionInfo();

      // Metrics (admin only)
      case path === '/metrics':
        return await handleMetrics(event, requestLogger);

      default:
        throw new NotFoundError(`Route ${event.httpMethod} ${event.path}`);
    }
  } catch (error) {
    return handleError(error, requestLogger);
  }
}

/**
 * Basic health check
 */
async function handleHealthCheck(logger: Logger): Promise<APIGatewayProxyResult> {
  const config = getConfig();

  return success({
    status: 'healthy',
    timestamp: new Date().toISOString(),
    environment: config.ENVIRONMENT,
    tier: config.TIER,
  });
}

/**
 * Deep health check (ready probe)
 */
async function handleReadyCheck(logger: Logger): Promise<APIGatewayProxyResult> {
  const config = getConfig();
  const checks: Record<string, { status: string; latency?: number }> = {};
  const startTime = Date.now();

  // Check database
  try {
```

```
    const dbStart = Date.now();
    await query('SELECT 1', {}, logger);
    checks.database = {
      status: 'healthy',
      latency: Date.now() - dbStart,
    };
  } catch (error) {
    checks.database = { status: 'unhealthy' };
    logger.error('Database health check failed', error as Error);
  }

  // Check LiteLLM
  try {
    const llmStart = Date.now();
    const health = await checkLiteLLMHealth(logger);
    checks.litellm = {
      status: health.status,
      latency: Date.now() - llmStart,
    };
  } catch (error) {
    checks.litellm = { status: 'unhealthy' };
    logger.error('LiteLLM health check failed', error as Error);
  }

  // Determine overall status
  const allHealthy = Object.values(checks).every(c => c.status === 'healthy');
  const overallStatus = allHealthy ? 'ready' : 'degraded';

  logger.info('Ready check completed', {
    status: overallStatus,
    checks,
    totalLatency: Date.now() - startTime,
  });

  return success({
    status: overallStatus,
    timestamp: new Date().toISOString(),
    environment: config.ENVIRONMENT,
    tier: config.TIER,
    checks,
  }, allHealthy ? 200 : 503);
}

/**
 * Version information
 */
```

```typescript
function handleVersionInfo(): APIGatewayProxyResult {
  const config = getConfig();

  return success({
    version: '2.2.0',
    appId: config.APP_ID,
    environment: config.ENVIRONMENT,
    tier: config.TIER,
    apiVersion: 'v2',
    buildDate: '2024-12',
  });
}

/**
 * Metrics endpoint (admin only)
 */
async function handleMetrics(
  event: APIGatewayProxyEvent,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  // Metrics collection - use Section 12 MetricsCollector service
  // See Section 12.2 for implementation details
  // await metricsCollector.recordUsage({ ... });
  // This would aggregate data from CloudWatch, DynamoDB usage table, etc.

  return success({
    message: 'Metrics endpoint - coming in Prompt 5',
    timestamp: new Date().toISOString(),
  });
}
```

PART 7: CHAT LAMBDA

packages/infrastructure/lambda/api/chat.ts

```typescript
/**
 * Chat Lambda - AI completions handler
 *
 * Handles:
 * - Chat completions via LiteLLM
 * - Streaming responses
 * - PHI sanitization
 * - Usage tracking
 */
```

```typescript
import type {
  APIGatewayProxyEvent,
  APIGatewayProxyResult,
  Context,
} from 'aws-lambda';
import { z } from 'zod';
import { v4 as uuid } from 'uuid';
import { Logger } from '../shared/logger';
import { getConfig, getFeatureFlags } from '../shared/config';
import { success, handleError, formatSSE, streamingHeaders } from '../shared/response';
import { extractAuthContext, logAuthContext } from '../shared/auth';
import { ValidationError, NotFoundError } from '../shared/errors';
import {
  createChatCompletion,
  streamChatCompletion,
  calculateCost,
  ChatCompletionRequest,
  ChatMessage,
} from '../shared/litellm';
import { getModelByName, getTenantPhiConfig, recordUsage } from '../shared/db';
import { sanitizePHI, createMappingRecord, DEFAULT_PHI_CONFIG } from '../shared/phi';
import { DynamoDBClient } from '@aws-sdk/client-dynamodb';
import { DynamoDBDocumentClient, PutCommand } from '@aws-sdk/lib-dynamodb';

// Initialize clients
const ddbClient = DynamoDBDocumentClient.from(new DynamoDBClient({}));
const logger = new Logger({ handler: 'chat' });

// Request validation schema
const chatRequestSchema = z.object({
  model: z.string().min(1),
  messages: z.array(z.object({
    role: z.enum(['system', 'user', 'assistant', 'function', 'tool']),
    content: z.union([
      z.string(),
      z.array(z.object({
        type: z.enum(['text', 'image_url']),
        text: z.string().optional(),
        image_url: z.object({
          url: z.string(),
          detail: z.enum(['low', 'high', 'auto']).optional(),
        }).optional(),
      })),
    ]),
    name: z.string().optional(),
  })).min(1),
```

```
  max_tokens: z.number().int().positive().max(128000).optional(),
  temperature: z.number().min(0).max(2).optional(),
  stream: z.boolean().optional().default(false),
  session_id: z.string().uuid().optional(),
  enable_phi: z.boolean().optional(),
  phi_categories: z.array(z.string()).optional(),
});

type ChatRequest = z.infer<typeof chatRequestSchema>;

/**
 * Main handler
 */
export async function handler(
  event: APIGatewayProxyEvent,
  context: Context
): Promise<APIGatewayProxyResult> {
  const requestLogger = logger.child({
    requestId: context.awsRequestId,
    path: event.path,
  });

  try {
    // Extract authentication
    const auth = extractAuthContext(event);
    logAuthContext(auth, requestLogger);
    requestLogger.setTenantId(auth.tenantId);
    requestLogger.setUserId(auth.userId);

    // Parse and validate request
    const body = event.body ? JSON.parse(event.body) : {};
    const parseResult = chatRequestSchema.safeParse(body);

    if (!parseResult.success) {
      throw new ValidationError(
        'Invalid request body',
        parseResult.error.flatten().fieldErrors as Record<string, string[]>
      );
    }

    const request = parseResult.data;
    const config = getConfig();
    const features = getFeatureFlags(config.TIER);

    // Get model information
    const model = await getModelByName(request.model, requestLogger);
```

```
if (!model) {
  throw new NotFoundError(`Model ${request.model}`);
}

// Get tenant PHI config
let phiConfig = DEFAULT_PHI_CONFIG;
if (features.phiSanitization && request.enable_phi !== false) {
  try {
    const tenantConfig = await getTenantPhiConfig(auth.tenantId, requestLogger);
    if (tenantConfig) {
      phiConfig = tenantConfig;
    }
  } catch {
    // Use default if tenant config not found
  }
}

// Sanitize PHI in messages
const sanitizedMessages = await sanitizeMessages(
  request.messages,
  phiConfig,
  auth.tenantId,
  request.session_id || null,
  requestLogger
);

// Build LiteLLM request
const litellmRequest: ChatCompletionRequest = {
  model: model.name,
  messages: sanitizedMessages.messages,
  max_tokens: request.max_tokens,
  temperature: request.temperature,
  stream: request.stream,
  user: auth.userId,
  metadata: {
    tenant_id: auth.tenantId,
    session_id: request.session_id,
    app_id: config.APP_ID,
  },
};

// Handle streaming vs non-streaming
if (request.stream) {
  return await handleStreamingRequest(
    litellmRequest,
    model,
```

```
        auth,
        request,
        sanitizedMessages.mappingId,
        requestLogger
      );
    } else {
      return await handleNonStreamingRequest(
        litellmRequest,
        model,
        auth,
        request,
        sanitizedMessages.mappingId,
        requestLogger
      );
    }
  } catch (error) {
    return handleError(error, requestLogger);
  }
}

/**
 * Sanitize PHI in messages
 */
async function sanitizeMessages(
  messages: ChatRequest['messages'],
  phiConfig: typeof DEFAULT_PHI_CONFIG,
  tenantId: string,
  sessionId: string | null,
  logger: Logger
): Promise<{ messages: ChatMessage[]; mappingId: string }> {
  if (phiConfig.mode === 'disabled') {
    return {
      messages: messages as ChatMessage[],
      mappingId: '',
    };
  }

  const config = getConfig();
  const sanitizedMessages: ChatMessage[] = [];
  let combinedMappingId = '';
  const allMappings: Record<string, string> = {};

  for (const message of messages) {
    if (typeof message.content === 'string') {
      const result = sanitizePHI(message.content, phiConfig, logger);
```

```
    if (result.matches.length > 0) {
      combinedMappingId = combinedMappingId || result.mappingId;

      // Collect mappings
      for (const match of result.matches) {
        allMappings[match.placeholder] = match.original;
      }
    }

    sanitizedMessages.push({
      ...message,
      content: result.sanitizedText,
    } as ChatMessage);
  } else {
    // Handle content parts (images, etc.)
    const sanitizedParts = [];

    for (const part of message.content) {
      if (part.type === 'text' && part.text) {
        const result = sanitizePHI(part.text, phiConfig, logger);

        if (result.matches.length > 0) {
          combinedMappingId = combinedMappingId || result.mappingId;
          for (const match of result.matches) {
            allMappings[match.placeholder] = match.original;
          }
        }

        sanitizedParts.push({
          ...part,
          text: result.sanitizedText,
        });
      } else {
        sanitizedParts.push(part);
      }
    }

    sanitizedMessages.push({
      ...message,
      content: sanitizedParts,
    } as ChatMessage);
  }
}

// Store PHI mapping in DynamoDB for re-identification
if (combinedMappingId && Object.keys(allMappings).length > 0) {
```

```
    const ttlHours = phiConfig.reidentification.mapping_ttl_hours;
    const expiresAt = Math.floor(Date.now() / 1000) + (ttlHours * 60 * 60);

    await ddbClient.send(new PutCommand({
      TableName: config.CACHE_TABLE,
      Item: {
        pk: `phi#${combinedMappingId}`,
        tenant_id: tenantId,
        session_id: sessionId,
        mappings: allMappings,
        created_at: new Date().toISOString(),
        ttl: expiresAt,
      },
    }));

    logger.info('PHI mappings stored', {
      mappingId: combinedMappingId,
      mappingCount: Object.keys(allMappings).length,
    });
  }

  return {
    messages: sanitizedMessages,
    mappingId: combinedMappingId,
  };
}

/**
 * Handle non-streaming request
 */
async function handleNonStreamingRequest(
  request: ChatCompletionRequest,
  model: Awaited<ReturnType<typeof getModelByName>>,
  auth: ReturnType<typeof extractAuthContext>,
  originalRequest: ChatRequest,
  phiMappingId: string,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  const startTime = Date.now();

  // Call LiteLLM
  const response = await createChatCompletion(request, logger);
  const latency = Date.now() - startTime;

  // Calculate costs
  const pricing = model!.pricing;
```

```
const { cost, billed } = calculateCost(response.usage, {
  input_tokens: pricing.input_tokens || 0,
  output_tokens: pricing.output_tokens || 0,
  billed_markup: pricing.billed_markup,
});

// Record usage
await recordUsage({
  tenant_id: auth.tenantId,
  user_id: auth.userId,
  session_id: originalRequest.session_id || null,
  model_id: model!.id,
  provider_id: model!.provider_id,
  input_tokens: response.usage.prompt_tokens,
  output_tokens: response.usage.completion_tokens,
  total_tokens: response.usage.total_tokens,
  cost,
  billed_amount: billed,
  request_type: 'chat.completion',
  latency_ms: latency,
}, logger);

// Store in sessions table if session_id provided
if (originalRequest.session_id) {
  const config = getConfig();
  await ddbClient.send(new PutCommand({
    TableName: config.SESSIONS_TABLE,
    Item: {
      pk: originalRequest.session_id,
      gsi1pk: auth.userId,
      gsi1sk: new Date().toISOString(),
      messages: [
        ...originalRequest.messages,
        response.choices[0]?.message,
      ],
      model: request.model,
      tenant_id: auth.tenantId,
      phi_mapping_id: phiMappingId || undefined,
      created_at: new Date().toISOString(),
      updated_at: new Date().toISOString(),
      ttl: Math.floor(Date.now() / 1000) + (30 * 24 * 60 * 60), // 30 days
    },
  }));
}

logger.info('Chat completion successful', {
```

```
    model: request.model,
    inputTokens: response.usage.prompt_tokens,
    outputTokens: response.usage.completion_tokens,
    latency,
    cost,
  });

  return success({
    id: response.id,
    object: response.object,
    created: response.created,
    model: response.model,
    choices: response.choices,
    usage: response.usage,
    session_id: originalRequest.session_id,
    phi_mapping_id: phiMappingId || undefined,
  });
}

/**
 * Handle streaming request
 * Note: API Gateway doesn't support true streaming, so we buffer and return
 * For true streaming, use WebSocket API or Lambda Function URLs
 */
async function handleStreamingRequest(
  request: ChatCompletionRequest,
  model: Awaited<ReturnType<typeof getModelByName>>,
  auth: ReturnType<typeof extractAuthContext>,
  originalRequest: ChatRequest,
  phiMappingId: string,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  const startTime = Date.now();
  const chunks: string[] = [];
  let totalContent = '';
  let finishReason: string | null = null;
  let usage = { prompt_tokens: 0, completion_tokens: 0, total_tokens: 0 };

  try {
    for await (const chunk of streamChatCompletion(request, logger)) {
      chunks.push(formatSSE(chunk));

      if (chunk.choices[0]?.delta?.content) {
        totalContent += chunk.choices[0].delta.content;
      }
```

```
    if (chunk.choices[0]?.finish_reason) {
      finishReason = chunk.choices[0].finish_reason;
    }

    if (chunk.usage) {
      usage = chunk.usage;
    }
  }

  // Add done marker
  chunks.push(formatSSE('[DONE]'));
} catch (error) {
  logger.error('Streaming error', error as Error);
  throw error;
}

const latency = Date.now() - startTime;

// Record usage if we have it
if (usage.total_tokens > 0) {
  const pricing = model!.pricing;
  const { cost, billed } = calculateCost(usage, {
    input_tokens: pricing.input_tokens || 0,
    output_tokens: pricing.output_tokens || 0,
    billed_markup: pricing.billed_markup,
  });

  await recordUsage({
    tenant_id: auth.tenantId,
    user_id: auth.userId,
    session_id: originalRequest.session_id || null,
    model_id: model!.id,
    provider_id: model!.provider_id,
    input_tokens: usage.prompt_tokens,
    output_tokens: usage.completion_tokens,
    total_tokens: usage.total_tokens,
    cost,
    billed_amount: billed,
    request_type: 'chat.completion.stream',
    latency_ms: latency,
  }, logger);
}

logger.info('Streaming completion successful', {
  model: request.model,
  chunkCount: chunks.length,
```

```
    contentLength: totalContent.length,
    latency,
  });

  // Return SSE formatted response
  // Note: For true streaming, implement Lambda Function URL or WebSocket
  return {
    statusCode: 200,
    headers: streamingHeaders(),
    body: chunks.join(''),
  };
}
```

PART 8: MODELS LAMBDA

packages/infrastructure/lambda/api/models.ts

```
/**
 * Models Lambda - Dynamic model registry
 *
 * Handles:
 * - List all models
 * - Get model by ID
 * - Filter by specialty, provider, status
 * - Admin: Update model status, thermal state
 */

import type {
  APIGatewayProxyEvent,
  APIGatewayProxyResult,
  Context,
} from 'aws-lambda';
import { z } from 'zod';
import { Logger } from '../shared/logger';
import { getConfig } from '../shared/config';
import { success, handleError, noContent } from '../shared/response';
import { extractAuthContext, requireAdmin, logAuthContext } from '../shared/auth';
import { ValidationError, NotFoundError } from '../shared/errors';
import { getModels, getModelById, updateModel as dbUpdateModel } from '../shared/db';
import { createAuditLog } from '../shared/db';

// Initialize logger
const logger = new Logger({ handler: 'models' });

// Query parameters schema
```

```typescript
const listQuerySchema = z.object({
  provider_id: z.string().optional(),
  specialty: z.string().optional(),
  status: z.enum(['active', 'inactive', 'deprecated', 'coming_soon']).optional(),
  type: z.enum(['external', 'self-hosted']).optional(),
  supports_vision: z.string().transform(v => v === 'true').optional(),
  supports_streaming: z.string().transform(v => v === 'true').optional(),
  limit: z.string().transform(Number).pipe(z.number().int().min(1).max(100)).optional(),
  offset: z.string().transform(Number).pipe(z.number().int().min(0)).optional(),
});

// Update request schema (admin only)
const updateModelSchema = z.object({
  status: z.enum(['active', 'inactive', 'deprecated', 'coming_soon']).optional(),
  thermal_state: z.enum(['OFF', 'COLD', 'WARM', 'HOT', 'AUTOMATIC']).optional(),
  display_name: z.string().min(1).max(200).optional(),
  description: z.string().max(2000).optional(),
});

/**
 * Main handler
 */
export async function handler(
  event: APIGatewayProxyEvent,
  context: Context
): Promise<APIGatewayProxyResult> {
  const requestLogger = logger.child({
    requestId: context.awsRequestId,
    path: event.path,
    method: event.httpMethod,
  });

  try {
    // Extract authentication
    const auth = extractAuthContext(event);
    logAuthContext(auth, requestLogger);

    // Parse path parameters
    const modelId = event.pathParameters?.modelId;

    switch (event.httpMethod) {
      case 'GET':
        if (modelId) {
          return await handleGetModel(modelId, requestLogger);
        }
        return await handleListModels(event, requestLogger);
```

```
      case 'PUT':
      case 'PATCH':
        if (!modelId) {
          throw new ValidationError('Model ID required for update');
        }
        requireAdmin(auth);
        return await handleUpdateModel(modelId, event, auth, requestLogger);

      default:
        throw new ValidationError(`Method ${event.httpMethod} not allowed`);
    }
  } catch (error) {
    return handleError(error, requestLogger);
  }
}

/**
 * List models with filtering
 */
async function handleListModels(
  event: APIGatewayProxyEvent,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  // Parse and validate query parameters
  const queryResult = listQuerySchema.safeParse(event.queryStringParameters || {});

  if (!queryResult.success) {
    throw new ValidationError(
      'Invalid query parameters',
      queryResult.error.flatten().fieldErrors as Record<string, string[]>
    );
  }

  const filters = queryResult.data;

  // Query database
  const result = await getModels({
    providerId: filters.provider_id,
    specialty: filters.specialty,
    status: filters.status,
    type: filters.type,
    supportsVision: filters.supports_vision,
    supportsStreaming: filters.supports_streaming,
    limit: filters.limit || 50,
    offset: filters.offset || 0,
```

```
  }, logger);

  // Transform to API response format
  const models = result.rows.map(transformModel);

  logger.info('Models listed', {
    count: models.length,
    filters,
  });

  return success({
    models,
    pagination: {
      limit: filters.limit || 50,
      offset: filters.offset || 0,
      total: result.rowCount,
      hasMore: result.rowCount > (filters.offset || 0) + models.length,
    },
  });
}

/**
 * Get single model by ID
 */
async function handleGetModel(
  modelId: string,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  const model = await getModelById(modelId, logger);

  logger.info('Model retrieved', { modelId });

  return success({
    model: transformModel(model),
  });
}

/**
 * Update model (admin only)
 */
async function handleUpdateModel(
  modelId: string,
  event: APIGatewayProxyEvent,
  auth: ReturnType<typeof extractAuthContext>,
  logger: Logger
): Promise<APIGatewayProxyResult> {
```

```
// Parse and validate request body
const body = event.body ? JSON.parse(event.body) : {};
const parseResult = updateModelSchema.safeParse(body);

if (!parseResult.success) {
  throw new ValidationError(
    'Invalid request body',
    parseResult.error.flatten().fieldErrors as Record<string, string[]>
  );
}

const updates = parseResult.data;

if (Object.keys(updates).length === 0) {
  throw new ValidationError('No updates provided');
}

// Get current model
const currentModel = await getModelById(modelId, logger);

// Update model
const updatedModel = await dbUpdateModel(modelId, updates, logger);

// Create audit log
await createAuditLog({
  tenant_id: auth.tenantId,
  user_id: null,
  admin_id: auth.userId,
  action: 'model.update',
  resource_type: 'model',
  resource_id: modelId,
  details: {
    before: {
      status: currentModel.status,
      thermal_state: currentModel.thermal_state,
      display_name: currentModel.display_name,
    },
    after: updates,
  },
  ip_address: event.requestContext.identity?.sourceIp || null,
  user_agent: event.headers['User-Agent'] || null,
}, logger);

logger.info('Model updated', {
  modelId,
  updates,
```

```
    adminId: auth.userId,
  });

  return success({
    model: transformModel(updatedModel),
  });
}

/**
 * Transform database model to API response format
 */
function transformModel(dbModel: any) {
  return {
    id: dbModel.id,
    providerId: dbModel.provider_id,
    name: dbModel.name,
    displayName: dbModel.display_name,
    description: dbModel.description,
    type: dbModel.type,
    specialty: dbModel.specialty,
    capabilities: dbModel.capabilities,
    contextWindow: dbModel.context_window,
    maxOutputTokens: dbModel.max_output_tokens,
    supportsFunctions: dbModel.supports_functions,
    supportsVision: dbModel.supports_vision,
    supportsStreaming: dbModel.supports_streaming,
    hasThinkingMode: dbModel.has_thinking_mode,
    thinkingBudgetTokens: dbModel.thinking_budget_tokens,
    pricing: {
      inputTokens: dbModel.pricing?.input_tokens,
      outputTokens: dbModel.pricing?.output_tokens,
      perImage: dbModel.pricing?.per_image,
      perMinuteAudio: dbModel.pricing?.per_minute_audio,
      perMinuteVideo: dbModel.pricing?.per_minute_video,
      per3DModel: dbModel.pricing?.per_3d_model,
      billedMarkup: dbModel.pricing?.billed_markup,
    },
    thermalState: dbModel.thermal_state,
    thermalConfig: dbModel.thermal_config,
    status: dbModel.status,
    releaseDate: dbModel.release_date,
    deprecationDate: dbModel.deprecation_date,
    createdAt: dbModel.created_at,
    updatedAt: dbModel.updated_at,
  };
}
```

PART 9: PROVIDERS LAMBDA

packages/infrastructure/lambda/api/providers.ts

```typescript
/**
 * Providers Lambda - AI provider management
 *
 * Handles:
 * - List all providers
 * - Get provider by ID
 * - Get provider models
 * - Admin: Update provider status
 */

import type {
  APIGatewayProxyEvent,
  APIGatewayProxyResult,
  Context,
} from 'aws-lambda';
import { z } from 'zod';
import { Logger } from '../shared/logger';
import { getConfig } from '../shared/config';
import { success, handleError } from '../shared/response';
import { extractAuthContext, requireAdmin, logAuthContext } from '../shared/auth';
import { ValidationError } from '../shared/errors';
import {
  getProviders,
  getProviderById,
  updateProvider as dbUpdateProvider,
  getModels,
  createAuditLog,
} from '../shared/db';

// Initialize logger
const logger = new Logger({ handler: 'providers' });

// Query parameters schema
const listQuerySchema = z.object({
  type: z.enum(['external', 'self-hosted', 'mid-tier']).optional(),
  status: z.enum(['active', 'inactive', 'deprecated']).optional(),
  hipaa_compliant: z.string().transform(v => v === 'true').optional(),
  limit: z.string().transform(Number).pipe(z.number().int().min(1).max(100)).optional(),
  offset: z.string().transform(Number).pipe(z.number().int().min(0)).optional(),
});
```

```typescript
// Update request schema (admin only)
const updateProviderSchema = z.object({
  status: z.enum(['active', 'inactive', 'deprecated']).optional(),
  hipaa_compliant: z.boolean().optional(),
  config: z.record(z.unknown()).optional(),
});

/**
 * Main handler
 */
export async function handler(
  event: APIGatewayProxyEvent,
  context: Context
): Promise<APIGatewayProxyResult> {
  const requestLogger = logger.child({
    requestId: context.awsRequestId,
    path: event.path,
    method: event.httpMethod,
  });

  try {
    // Extract authentication
    const auth = extractAuthContext(event);
    logAuthContext(auth, requestLogger);

    // Parse path
    const providerId = event.pathParameters?.providerId;
    const subPath = event.path.split('/').pop();

    switch (event.httpMethod) {
      case 'GET':
        if (providerId) {
          if (subPath === 'models') {
            return await handleGetProviderModels(providerId, requestLogger);
          }
          return await handleGetProvider(providerId, requestLogger);
        }
        return await handleListProviders(event, requestLogger);

      case 'PUT':
      case 'PATCH':
        if (!providerId) {
          throw new ValidationError('Provider ID required for update');
        }
        requireAdmin(auth);
```

```
        return await handleUpdateProvider(providerId, event, auth, requestLogger);

      default:
        throw new ValidationError(`Method ${event.httpMethod} not allowed`);
    }
  } catch (error) {
    return handleError(error, requestLogger);
  }
}

/**
 * List providers with filtering
 */
async function handleListProviders(
  event: APIGatewayProxyEvent,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  // Parse and validate query parameters
  const queryResult = listQuerySchema.safeParse(event.queryStringParameters || {});

  if (!queryResult.success) {
    throw new ValidationError(
      'Invalid query parameters',
      queryResult.error.flatten().fieldErrors as Record<string, string[]>
    );
  }

  const filters = queryResult.data;

  // Query database
  const result = await getProviders({
    type: filters.type,
    status: filters.status,
    hipaaCompliant: filters.hipaa_compliant,
    limit: filters.limit || 50,
    offset: filters.offset || 0,
  }, logger);

  // Transform to API response format
  const providers = result.rows.map(transformProvider);

  logger.info('Providers listed', {
    count: providers.length,
    filters,
  });
```

```
  return success({
    providers,
    pagination: {
      limit: filters.limit || 50,
      offset: filters.offset || 0,
      total: result.rowCount,
      hasMore: result.rowCount > (filters.offset || 0) + providers.length,
    },
  });
}

/**
 * Get single provider by ID
 */
async function handleGetProvider(
  providerId: string,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  const provider = await getProviderById(providerId, logger);

  logger.info('Provider retrieved', { providerId });

  return success({
    provider: transformProvider(provider),
  });
}

/**
 * Get models for a provider
 */
async function handleGetProviderModels(
  providerId: string,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  // Verify provider exists
  await getProviderById(providerId, logger);

  // Get provider's models
  const result = await getModels({
    providerId,
    status: 'active',
  }, logger);

  const models = result.rows.map(model => ({
    id: model.id,
    name: model.name,
```

```
    displayName: model.display_name,
    specialty: model.specialty,
    status: model.status,
    thermalState: model.thermal_state,
  }));

  logger.info('Provider models retrieved', {
    providerId,
    modelCount: models.length,
  });

  return success({
    providerId,
    models,
  });
}

/**
 * Update provider (admin only)
 */
async function handleUpdateProvider(
  providerId: string,
  event: APIGatewayProxyEvent,
  auth: ReturnType<typeof extractAuthContext>,
  logger: Logger
): Promise<APIGatewayProxyResult> {
  // Parse and validate request body
  const body = event.body ? JSON.parse(event.body) : {};
  const parseResult = updateProviderSchema.safeParse(body);

  if (!parseResult.success) {
    throw new ValidationError(
      'Invalid request body',
      parseResult.error.flatten().fieldErrors as Record<string, string[]>
    );
  }

  const updates = parseResult.data;

  if (Object.keys(updates).length === 0) {
    throw new ValidationError('No updates provided');
  }

  // Get current provider
  const currentProvider = await getProviderById(providerId, logger);
```

```
  // Update provider
  const updatedProvider = await dbUpdateProvider(providerId, {
    status: updates.status,
    hipaa_compliant: updates.hipaa_compliant,
    config: updates.config,
  }, logger);

  // Create audit log
  await createAuditLog({
    tenant_id: auth.tenantId,
    user_id: null,
    admin_id: auth.userId,
    action: 'provider.update',
    resource_type: 'provider',
    resource_id: providerId,
    details: {
      before: {
        status: currentProvider.status,
        hipaa_compliant: currentProvider.hipaa_compliant,
      },
      after: updates,
    },
    ip_address: event.requestContext.identity?.sourceIp || null,
    user_agent: event.headers['User-Agent'] || null,
  }, logger);

  logger.info('Provider updated', {
    providerId,
    updates,
    adminId: auth.userId,
  });

  return success({
    provider: transformProvider(updatedProvider),
  });
}

/**
 * Transform database provider to API response format
 */
function transformProvider(dbProvider: any) {
  return {
    id: dbProvider.id,
    name: dbProvider.name,
    type: dbProvider.type,
    status: dbProvider.status,
```

```
    hipaaCompliant: dbProvider.hipaa_compliant,
    baaAvailable: dbProvider.baa_available,
    baseUrl: dbProvider.base_url,
    authType: dbProvider.auth_type,
    capabilities: dbProvider.capabilities,
    createdAt: dbProvider.created_at,
    updatedAt: dbProvider.updated_at,
  };
}
```

DEPLOYMENT VERIFICATION

After deploying the Lambda functions, verify they work:

```
# 1. Health check
curl https://api.thinktank.YOUR_DOMAIN.com/api/v2/health

# Expected response:
# {
#   "success": true,
#   "data": {
#     "status": "healthy",
#     "timestamp": "2024-12-21T...",
#     "environment": "dev",
#     "tier": 1
#   }
# }

# 2. List models (requires auth token)
curl -H "Authorization: Bearer $TOKEN" \
  https://api.thinktank.YOUR_DOMAIN.com/api/v2/models

# 3. List providers
curl -H "Authorization: Bearer $TOKEN" \
  https://api.thinktank.YOUR_DOMAIN.com/api/v2/providers

# 4. Chat completion
curl -X POST \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "gpt-4",
    "messages": [{"role": "user", "content": "Hello!"}]
  }' \
  https://api.thinktank.YOUR_DOMAIN.com/api/v2/chat/completions
```

NEXT PROMPTS

Continue with: - **Prompt 5**: Lambda Functions - Admin & Billing (Invitations, Approvals, Metering) - **Prompt 6**: Self-Hosted Models & Mid-Level Services Configuration - **Prompt 7**: External Providers & Database Schema/Migrations - **Prompt 8**: Admin Web Dashboard (Next.js) - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 4: Lambda Functions - Core RADIANT v2.2.0 - December 2024

â• â•

END OF SECTION 4

â• â•

â• â•

SECTION-05-LAMBDA-ADMIN-BILLING

SECTION 5: LAMBDA FUNCTIONS - ADMIN & BILLING (v2.1.0)

â• â•

> **Dependencies:** Sections 0-4 **Creates:** Invitations, Approvals, Metering, Billing handlers

Type Imports

```
import { Administrator, AdminRole, Invitation, ApprovalRequest, Invoice } from '@radiant/sha
```

> **IMPORTANT:** Use canonical table names: administrators, invitations, approval_requests

---

RADIANT v2.2.0 - Prompt 5: Lambda Functions - Admin & Billing

Prompt 5 of 9 | Target Size: ~60KB | Version: 3.7.0 | December 2024

OVERVIEW

This prompt creates the Admin & Billing Lambda functions for the RADIANT platform:

1. **Invitations Lambda** - Email-based administrator invitations with secure tokens
2. **Approvals Lambda** - Two-person approval workflow for production deployments
3. **Admin Users Lambda** - Administrator CRUD, roles, and MFA management
4. **Admin Profiles Lambda** - Preferences, notifications, and settings
5. **Metering Lambda** - Real-time usage event collection and tracking
6. **Billing Lambda** - Cost aggregation, invoicing, and payment processing
7. **Audit Lambda** - Comprehensive audit logging and compliance reporting
8. **Notifications Lambda** - Admin notification delivery and management

---

KEY FEATURES

Administrator Management

- **Email Invitations**: Secure token-based invitation system with expiry
- **Role-Based Access**: super_admin, admin, operator, auditor roles
- **MFA Requirement**: Mandatory MFA for production environment access
- **Two-Person Approval**: Production deployments require separate initiator and approver

Billing & Metering

- **Real-Time Tracking**: Every API call metered with token counts and costs
- **Dynamic Pricing**: Costs pulled from Dynamic Model Registry
- **Configurable Margin**: Default 20% markup on provider costs
- **Stripe Integration**: Automatic billing and payment processing
- **Usage Rollups**: Daily aggregation for efficient querying

---

LAMBDA DIRECTORY STRUCTURE

“‘ packages/infrastructure/lambda/ Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ admin/ Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ invitations.ts # Email invitations Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ approvals.ts # Two-person approval workflow Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ users.ts # Admin user CRUD Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ profiles.ts # Admin preferences Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ notifications.ts # Notification management Ã¢â€ â€š Ã¢â€ â€š Ã¢â€ â‚¬Ã¢â€ â‚¬ audit.ts # Audit log queries Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ billing/ Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ metering.ts # Usage event collection Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ billing.ts # Cost aggregation & invoices Ã¢â€ â€š Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ pricing.ts # Dynamic pricing sync Ã¢â€ â€š Ã¢â€ â€š Ã¢â€ â‚¬Ã¢â€ â‚¬ payments.ts # Stripe integration Ã¢â€ â€š Ã¢â€ â‚¬Ã¢â€ â‚¬ shared/ Ã¢â€ â€š Ã¢â€ â‚¬Ã¢â€ â‚¬ admin/ Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ types.ts # Admin-specific types Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ email.ts # SES email utilities Ã¢â€ Å“Ã¢â€ â‚¬Ã¢â€ â‚¬ tokens.ts # Secure token generation Ã¢â€ â€š Ã¢â€ â‚¬Ã¢â€ â‚¬ stripe.ts # Stripe client “‘

PART 1: SHARED ADMIN UTILITIES

packages/infrastructure/lambda/shared/admin/index.ts

“‘typescript // Re-export admin utilities export * from ‘./types’; export * from ‘./email’; export * from ‘./tokens’; export * from ‘./stripe’; “‘

packages/infrastructure/lambda/shared/admin/types.ts

“‘typescript /** * Admin-specific types for RADIANT v2.2.0 */

import { z } from ‘zod’;

// ==========
// ADMINISTRATOR ROLES // ==========

export const AdminRole = { SUPER_ADMIN: ‘super_admin’, ADMIN: ‘admin’, OPERATOR: ‘operator’, AUDITOR: ‘auditor’, } as const;

export type AdminRoleType = typeof AdminRole[keyof typeof AdminRole];

export const ROLE_HIERARCHY: Record<AdminRoleType, number> = { [AdminRole.SUPER_ADMIN]: 100, [AdminRole.ADMIN]: 75, [AdminRole.OPERATOR]: 50, [AdminRole.AUDITOR]: 25, };

export const ROLE_PERMISSIONS: Record<AdminRoleType, string[]> = { [AdminRole.SUPER_ADMIN]: [‘admin:’, ’billing:’, ‘settings:’, ’deployments:’, ‘approvals:’, ’audit:’,], [AdminRole.ADMIN]: [‘admin:read’, ‘admin:write’,

‘billing:read’, ‘settings:read’, ‘settings:write’, ‘deployments:read’, ‘deployments:write’, ‘approvals:read’, ‘approvals:initiate’,], [AdminRole.OPERATOR]: [‘admin:read’, ‘billing:read’, ‘settings:read’, ‘deployments:read’,], [AdminRole.AUDITOR]: [‘admin:read’, ‘billing:read’, ‘audit:read’,], };

// ==
// INVITATION TYPES // ==

export interface Invitation { id: string; email: string; role: AdminRoleType; invitedBy: string; invitedByName: string; appId: string; tenantId: string; environment: ‘dev’ | ‘staging’ | ‘prod’; token: string; tokenHash: string; expiresAt: string; status: ‘pending’ | ‘accepted’ | ‘expired’ | ‘revoked’; createdAt: string; acceptedAt?: string; acceptedByIp?: string; }

export const createInvitationSchema = z.object({ email: z.string().email(), role: z.enum([‘super_admin’, ‘admin’, ‘operator’, ‘auditor’]), message: z.string().max(500).optional(), expiresInHours: z.number().int().min(1).max(168).default(48), });

export const acceptInvitationSchema = z.object({ token: z.string().min(32), firstName: z.string().min(1).max(50), lastName: z.string().min(1).max(50), password: z.string().min(12).max(128), mfaMethod: z.enum([‘authenticator’, ‘sms’, ‘email’]).default(‘authenticator’), phone: z.string().optional(), });

// ==
// APPROVAL TYPES // ==

export type ApprovalType = | ‘deployment’ | ‘promotion’ | ‘model_activation’ | ‘provider_change’ | ‘user_role_change’ | ‘billing_change’;

export type ApprovalStatus = | ‘pending’ | ‘approved’ | ‘rejected’ | ‘expired’ | ‘cancelled’;

export interface ApprovalRequest { id: string; type: ApprovalType; appId: string; tenantId: string; environment: ‘dev’ | ‘staging’ | ‘prod’; requestedBy: string; requestedByName: string; requestedAt: string; expiresAt: string; status: ApprovalStatus; approvedBy?: string; approvedByName?: string; approvedAt?: string; rejectedReason?: string; action: string; resourceType: string; resourceId: string; details: Record<string, unknown>; priority: ‘low’ | ‘medium’ | ‘high’ | ‘critical’; notes?: string; }

export const createApprovalSchema = z.object({ type: z.enum([‘deployment’, ‘promotion’, ‘model_activation’, ‘provider_change’, ‘user_role_change’, ‘billing_change’,]), action: z.string(), resourceType: z.string(), resourceId: z.string(), details: z.record(z.unknown()), priority: z.enum([‘low’, ‘medium’, ‘high’, ‘critical’]).default(‘medium’), notes: z.string().max(1000).optional(), expiresInHours: z.number().int().min(1).max(168).default(24), });

export const processApprovalSchema = z.object({ action: z.enum([‘approve’, ‘reject’]), reason: z.string().max(500).optional(), }); “‘

packages/infrastructure/lambda/shared/admin/tokens.ts

“‘typescript /** * Secure token generation and validation */

import { randomBytes, createHash, timingSafeEqual } from ‘crypto’;

export function generateToken(length: number = 32): string { return randomBytes(length).toString(‘base64url’); }

export function generateCode(length: number = 6): string { const chars = ‘0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ’; const bytes = randomBytes(length); let code = ’’; for (let i = 0; i < length; i++) { code += chars[bytes[i] % chars.length]; } return code; }

export function hashToken(token: string): string { return createHash(‘sha256’).update(token).digest(‘hex’); }

export function verifyToken(token: string, hash: string): boolean { const tokenHash = hashToken(token); try { return timingSafeEqual(Buffer.from(tokenHash, ‘hex’), Buffer.from(hash, ‘hex’)); } catch { return false; } }

export function generateInvitationToken(): { token: string; tokenHash: string } { const token = generateToken(48); const tokenHash = hashToken(token); return { token, tokenHash }; }

export function calculateExpiry(hoursFromNow: number): string { const expiry = new Date(); expiry.setHours(expiry.getHours() + hoursFromNow); return expiry.toISOString(); }

export function isExpired(expiresAt: string): boolean { return new Date(expiresAt) < new Date(); } “‘

packages/infrastructure/lambda/shared/admin/email.ts

“‘typescript /** * Email utilities using AWS SES */

import { SESClient, SendEmailCommand } from ‘@aws-sdk/client-ses’; import { Logger } from ‘../logger’;

const sesClient = new SESClient({});

export interface EmailOptions { to: string | string[]; subject: string; html: string; text?: string; replyTo?: string; }

export async function sendEmail(options: EmailOptions, logger: Logger): Promise { const fromEmail = ‘noreply@${process.env.DOMAIN || ‘radiant.cloud’}‘; const toAddresses = Array.isArray(options.to) ? options.to : [options.to];

const command = new SendEmailCommand({ Source: fromEmail, Destination: { ToAddresses: toAddresses }, Message: { Subject: { Data: options.subject, Charset: ‘UTF-8’ }, Body: { Html: { Data: options.html, Charset: ‘UTF-8’ },

…(options.text && { Text: { Data: options.text, Charset: ‘UTF-8’ } }), }, }, ReplyToAddresses: options.replyTo ? [options.replyTo] : undefined, });

try { await sesClient.send(command); logger.info(‘Email sent successfully’, { to: toAddresses, subject: options.subject }); } catch (error) { logger.error(‘Failed to send email’, error as Error, { to: toAddresses }); throw error; } }

export function generateInvitationEmail(params: { inviteeName: string; inviterName: string; role: string; appName: string; environment: string; acceptUrl: string; expiresAt: string; message?: string; }): { html: string; text: string } { const html = ‘ <!DOCTYPE html>

You’ve Been Invited to ${params.appName}

```
<h1 style="color: #1a1a1a; margin-bottom: 20px;">You're Invited!</h1>
<p><strong>\${params.inviterName}</strong> has invited you to join <strong>\${params.appName
\${params.message ? \`<blockquote style="background: #f8f9fa; padding: 15px; border-left: 4p
<div style="text-align: center; margin: 30px 0;">
  <a href="\${params.acceptUrl}" style="display: inline-block; padding: 14px 32px; backgroun
</div>
<p style="color: #666; font-size: 14px;">This invitation expires on <strong>\${new Date(para
```

‘;

const text = ‘You’ve been invited to ${params.appName} by ${params.inviterName} as a ${params.role}. Accept: ${params.acceptUrl}‘; return { html, text }; }

export function generateApprovalEmail(params: { approverName: string; requesterName: string; appName: string; environment: string; action: string; resourceType: string; resourceId: string; approveUrl: string; rejectUrl: string; expiresAt: string; }): { html: string; text: string } { const html = ‘ <!DOCTYPE html>

Approval Required - ${params.appName}

```
<div style="background: #ffc107; padding: 20px; text-align: center;">
  <h1 style="margin: 0; color: #1a1a1a;">Ã¢Å¡Ã Ã¯Ã¸Ã  Approval Required</h1>
</div>
<div style="padding: 30px;">
  <p>Hi \${params.approverName},</p>
  <p><strong>\${params.requesterName}</strong> has requested approval for:</p>
  <div style="background: #f8f9fa; padding: 20px; border-radius: 6px; margin: 20px 0;">
    <p><strong>Environment:</strong> \${params.environment.toUpperCase()}</p>
    <p><strong>Action:</strong> \${params.action}</p>
    <p><strong>Resource:</strong> \${params.resourceType} (\${params.resourceId})</p>
  </div>
  <div style="text-align: center; margin: 30px 0;">
    <a href="\${params.approveUrl}" style="display: inline-block; padding: 14px 32px; backgr
    <a href="\${params.rejectUrl}" style="display: inline-block; padding: 14px 32px; backgro
  </div>
```

```
  <p style="color: #dc3545; font-size: 12px; text-align: center;">Ã¢Å¡Ã Ã¯Ã¸Ã Production de
</div>
```

‘;

const text = ‘APPROVAL REQUIRED: ${params.requesterName} requests approval for ${params.action} on ${params.resourceType}. Approve: ${params.approveUrl} Reject: ${params.rejectUrl}‘; return { html, text }; } “‘

packages/infrastructure/lambda/shared/admin/stripe.ts

“‘typescript /** * Stripe payment integration */

import Stripe from ‘stripe’; import { SecretsManagerClient, GetSecretValueCommand } from ‘@aws-sdk/client-secrets-manager’; import { Logger } from ‘../logger’;

const secretsClient = new SecretsManagerClient({}); let stripeClient: Stripe | null = null;

async function getStripeClient(): Promise { if (stripeClient) return stripeClient;

const secretArn = process.env.STRIPE_SECRET_ARN; if (!secretArn) throw new Error(‘STRIPE_SECRET_ARN not configured’);

const command = new GetSecretValueCommand({ SecretId: secretArn }); const response = await secretsClient.send(command); if (!response.SecretString) throw new Error(‘Stripe secret not found’);

const secrets = JSON.parse(response.SecretString); stripeClient = new Stripe(secrets.apiKey, { apiVersion: ‘2023-10-16’ }); return stripeClient; }

export async function getOrCreateCustomer(tenantId: string, email: string, name: string, metadata: Record<string, string>, logger: Logger): Promise { const stripe = await getStripeClient();

const existing = await stripe.customers.search({ query: ‘metadata[‘tenantId’]:‘${tenantId}’‘, limit: 1, });

if (existing.data.length > 0) { logger.info(‘Found existing Stripe customer’, { tenantId, customerId: existing.data[0].id }); return existing.data[0].id; }

const customer = await stripe.customers.create({ email, name, metadata: { tenantId, …metadata }, });

logger.info(‘Created Stripe customer’, { tenantId, customerId: customer.id }); return customer.id; }

export async function createInvoice(params: { customerId: string; tenantId: string; periodStart: Date; periodEnd: Date; lineItems: Array<{ description: string; amount: number; quantity: number; metadata?: Record<string, string>; }>; metadata?: Record<string, string>; }, logger: Logger): Promise<Stripe.Invoice> { const stripe = await getStripeClient();

for (const item of params.lineItems) { await stripe.invoiceItems.create({ customer: params.customerId, amount: item.amount, currency: ‘usd’, description: item.description, quantity: item.quantity, metadata: item.metadata, }); }

const invoice = await stripe.invoices.create({ customer: params.customerId, auto_advance: true, collection_method: ‘charge_automatically’, metadata: { tenantId: params.tenantId, periodStart: params.periodStart.toISOString(), periodEnd: params.periodEnd.toISOString(), …params.metadata, }, });

logger.info(‘Created Stripe invoice’, { tenantId: params.tenantId, invoiceId: invoice.id }); return invoice; }

export async function getInvoiceStatus(invoiceId: string, logger: Logger): Promise<{ status: string; paid: boolean; amountDue: number; amountPaid: number; }> { const stripe = await getStripeClient(); const invoice = await stripe.invoices.retrieve(invoiceId); return { status: invoice.status || ‘unknown’, paid: invoice.paid, amountDue: invoice.amount_due, amountPaid: invoice.amount_paid, }; } “‘

PART 2: INVITATIONS LAMBDA

packages/infrastructure/lambda/admin/invitations.ts

“‘typescript /** * Administrator Invitation Lambda * Handles email-based administrator invitations with secure tokens */

import { APIGatewayProxyEvent, APIGatewayProxyResult, Context } from ‘aws-lambda’; import { v4 as uuidv4 } from ‘uuid’; import { Logger } from ‘../shared/logger’; import { success, created, handleError } from ‘../shared/response’; import { extractAuthContext, requireAdmin, requirePermission } from ‘../shared/auth’; import { ValidationError, NotFoundError, ForbiddenError } from ‘../shared/errors’; import { executeQuery, createAuditLog } from ‘../shared/db’; import { createInvitationSchema, acceptInvitationSchema, ROLE_HIERARCHY } from ‘../shared/admin/types’; import { generateInvitationToken, hashToken, calculateExpiry, isExpired } from ‘../shared/admin/tokens’; import { sendEmail, generateInvitationEmail } from ‘../shared/admin/email’; import { CognitoIdentityProviderClient, AdminCreateUserCommand, AdminAddUserToGroupCommand, AdminSetUserMFAPreferenceCommand, } from ‘@aws-sdk/client-cognito-identity-provider’;

const logger = new Logger({ handler: ‘invitations’ }); const cognitoClient = new CognitoIdentityProviderClient({});

export async function handler(event: APIGatewayProxyEvent, context: Context): Promise { const requestLogger = logger.child({ requestId: context.awsRequestId, path: event.path });

try { const invitationId = event.pathParameters?.invitationId; const action = event.path.split(‘/’).pop();

```
switch (event.httpMethod) {
  case 'GET':
    if (invitationId) return await handleGetInvitation(invitationId, event, requestLogger);
    return await handleListInvitations(event, requestLogger);

  case 'POST':
    if (action === 'accept') return await handleAcceptInvitation(event, requestLogger);
    if (action === 'resend' && invitationId) return await handleResendInvitation(invitationI
    return await handleCreateInvitation(event, requestLogger);

  case 'DELETE':
    if (!invitationId) throw new ValidationError('Invitation ID required');
    return await handleRevokeInvitation(invitationId, event, requestLogger);

  default:
    throw new ValidationError(\`Method \${event.httpMethod} not allowed\`);
}
```

} catch (error) { return handleError(error, requestLogger); } }

async function handleCreateInvitation(event: APIGatewayProxyEvent, logger: Logger): Promise { const auth = extractAuthContext(event); requireAdmin(auth); requirePermission(auth, ‘admin:write’);

const body = event.body ? JSON.parse(event.body) : {}; const parseResult = createInvitationSchema.safeParse(body); if (!parseResult.success) { throw new ValidationError(‘Invalid request body’, parseResult.error.flatten().fieldErrors as Record<string, string[]>); }

const { email, role, message, expiresInHours } = parseResult.data;

// Validate role hierarchy if (ROLE_HIERARCHY[role] > ROLE_HIERARCHY[auth.role as keyof typeof ROLE_HIERARCHY]) { throw new ForbiddenError(‘Cannot invite administrator with higher role than your own’); }

// Check existing admin/invitation const existingAdmin = await executeQuery(‘SELECT id FROM administrators WHERE email = $1 AND tenant_id = $2‘, [email, auth.tenantId], logger); if (existingAdmin.rowCount > 0) { throw new ValidationError(‘An administrator with this email already exists’); }

const pendingInvitation = await executeQuery(‘SELECT id FROM invitations WHERE email = $1 AND tenant_id = $2 AND status = ‘pending’ AND expires_at > NOW()‘, [email, auth.tenantId], logger); if (pendingInvitation.rowCount > 0) { throw new ValidationError(‘A pending invitation already exists for this email’); }

// Generate token and create invitation const { token, tokenHash } = gener-

ateInvitationToken(); const expiresAt = calculateExpiry(expiresInHours); const invitationId = uuidv4();

const inviterResult = await executeQuery(‘SELECT first_name, last_name FROM administrators WHERE id = $1‘, [auth.userId], logger); const inviterName = inviterResult.rows[0] ? ‘${inviterResult.rows[0].first_name} ${inviterResult.rows[0].last_name}‘ : ‘An administrator’;

const appResult = await executeQuery(‘SELECT name FROM apps WHERE id = $1‘, [auth.appId], logger); const appName = appResult.rows[0]?.name || auth.appId;

await executeQuery(‘INSERT INTO invitations (id, email, role, invited_by, app_id, tenant_id, environment, token_hash, expires_at, status, message, created_at) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, ‘pending’, $10, NOW())‘, [invitationId, email, role, auth.userId, auth.appId, auth.tenantId, auth.environment, tokenHash, expiresAt, message || null], logger);

// Send email const acceptUrl = ‘${process.env.ADMIN_URL}/invite/accept?token=${token}‘; const emailContent = generateInvitationEmail({ inviteeName: ’’, inviterName, role, appName, environment: auth.environment, acceptUrl, expiresAt, message, }); await sendEmail({ to: email, subject: ‘You’ve been invited to ${appName}‘, html: emailContent.html, text: emailContent.text }, logger);

await createAuditLog({ tenant_id: auth.tenantId, user_id: null, admin_id: auth.userId, action: ‘invitation.create’, resource_type: ‘invitation’, resource_id: invitationId, details: { email, role, expiresAt }, ip_address: event.requestContext.identity?.sourceIp || null, user_agent: event.headers[‘User-Agent’] || null, }, logger);

logger.info(‘Invitation created and sent’, { invitationId, email, role }); return created({ invitation: { id: invitationId, email, role, status: ‘pending’, expiresAt, createdAt: new Date().toISOString() } }); }

async function handleAcceptInvitation(event: APIGatewayProxyEvent, logger: Logger): Promise { const body = event.body ? JSON.parse(event.body) : {}; const parseResult = acceptInvitationSchema.safeParse(body); if (!parseResult.success) { throw new ValidationError(‘Invalid request body’, parseResult.error.flatten().fieldErrors as Record<string, string[]>); }

const { token, firstName, lastName, password, mfaMethod, phone } = parseResult.data;

const tokenHash = hashToken(token); const inviteResult = await executeQuery(‘SELECT * FROM invitations WHERE token_hash = $1 AND status = ‘pending‘, [tokenHash], logger); if (inviteResult.rowCount === 0) throw new NotFoundError(‘Invalid or expired invitation’);

const invitation = inviteResult.rows[0]; if (isExpired(invitation.expires_at)) { await executeQuery(‘UPDATE invitations SET status = ‘expired’ WHERE id

= $1‘, [invitation.id], logger); throw new ValidationError(‘This invitation has expired’); }

if (invitation.environment === ‘prod’ && mfaMethod === ‘sms’ && !phone) { throw new ValidationError(‘Phone number required for SMS MFA’); }

const userPoolId = process.env.ADMIN_USER_POOL_ID; const adminUserId = uuidv4();

// Create Cognito user await cognitoClient.send(new AdminCreateUserCommand({ UserPoolId: userPoolId, Username: invitation.email, TemporaryPassword: password, UserAttributes: [{ Name: ‘email’, Value: invitation.email }, { Name: ‘email_verified’, Value: ‘true’ }, { Name: ‘given_name’, Value: firstName }, { Name: ‘family_name’, Value: lastName }, { Name: ‘custom:adminId’, Value: adminUserId }, { Name: ‘custom:tenantId’, Value: invitation.tenant_id }, { Name: ‘custom:role’, Value: invitation.role },], MessageAction: ‘SUPPRESS’, }));

await cognitoClient.send(new AdminAddUserToGroupCommand({ UserPoolId: userPoolId, Username: invitation.email, GroupName: invitation.role, }));

if (invitation.environment === ‘prod’) { await cognitoClient.send(new AdminSetUserMFAPreferenceCommand({ UserPoolId: userPoolId, Username: invitation.email, SoftwareTokenMfaSettings: mfaMethod === ‘authenticator’ ? { Enabled: true, PreferredMfa: true } : undefined, SMSMfaSettings: mfaMethod === ‘sms’ ? { Enabled: true, PreferredMfa: true } : undefined, })); }

// Create admin user record await executeQuery(‘INSERT INTO administrators (id, cognito_user_id, email, first_name, last_name, display_name, role, app_id, tenant_id, mfa_enabled, mfa_method, status, created_at, updated_at, created_by, invitation_id) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, ‘active’, NOW(), NOW(), $12, $13)‘, [adminUserId, invitation.email, invitation.email, firstName, lastName, ‘${firstName} ${lastName}‘, invitation.role, invitation.app_id, invitation.tenant_id, invitation.environment === ‘prod’, mfaMethod, invitation.invited_by, invitation.id], logger);

// Create default profile await executeQuery(‘INSERT INTO admin_profiles (admin_id, notifications, timezone, language, date_format, time_format, currency, theme, default_environment, sidebar_collapsed, table_rows_per_page, updated_at) VALUES ($1, $2, ‘America/New_York’, ‘en’, ‘MM/DD/YYYY’, ‘12h’, ‘USD’, ‘system’, $3, false, 25, NOW())‘, [adminUserId, JSON.stringify({ method: ‘email’, frequency: ‘immediate’, categories: { security: true, billing: true, deployments: true, approvals: true, system: true } }), invitation.environment], logger);

await executeQuery(‘UPDATE invitations SET status = ‘accepted’, accepted_at = NOW(), accepted_by_ip = $2 WHERE id = $1‘, [invitation.id, event.requestContext.identity?.sourceIp || null], logger);

await createAuditLog({ tenant_id: invitation.tenant_id, user_id: null, ad-

min_id: adminUserId, action: 'invitation.accept', resource_type: 'invitation', resource_id: invitation.id, details: { email: invitation.email, role: invitation.role, firstName, lastName }, ip_address: event.requestContext.identity?.sourceIp || null, user_agent: event.headers['User-Agent'] || null, }, logger);

logger.info('Invitation accepted', { invitationId: invitation.id, adminUserId, email: invitation.email }); return success({ message: 'Invitation accepted successfully', adminUser: { id: adminUserId, email: invitation.email, firstName, lastName, role: invitation.role, mfaRequired: invitation.environment === 'prod' }, }); }

async function handleListInvitations(event: APIGatewayProxyEvent, logger: Logger): Promise { const auth = extractAuthContext(event); requireAdmin(auth); requirePermission(auth, 'admin:read');

const status = event.queryStringParameters?.status; const limit = parseInt(event.queryStringParameters?.limit || '50'); const offset = parseInt(event.queryStringParameters?.offset || '0');

let query = 'SELECT i.*, a.first_name as inviter_first_name, a.last_name as inviter_last_name FROM invitations i LEFT JOIN administrators a ON i.invited_by = a.id WHERE i.tenant_id = $1'; const params: any[] = [auth.tenantId];

if (status) { params.push(status); query += ' AND i.status = $${params.length}'; } query += ' ORDER BY i.created_at DESC LIMIT $${params.length + 1} OFFSET $${params.length + 2}'; params.push(limit, offset);

const result = await executeQuery(query, params, logger); const invitations = result.rows.map(row => ({ id: row.id, email: row.email, role: row.role, status: row.status, environment: row.environment, invitedBy: { id: row.invited_by, name: row.inviter_first_name ? '${row.inviter_first_name} ${row.inviter_last_name}' : 'Unknown' }, message: row.message, expiresAt: row.expires_at, createdAt: row.created_at, acceptedAt: row.accepted_at, }));

return success({ invitations, pagination: { limit, offset, hasMore: invitations.length === limit } }); }

async function handleGetInvitation(invitationId: string, event: APIGatewayProxyEvent, logger: Logger): Promise { const auth = extractAuthContext(event); requireAdmin(auth); requirePermission(auth, 'admin:read');

const result = await executeQuery('SELECT i.*, a.first_name as inviter_first_name, a.last_name as inviter_last_name FROM invitations i LEFT JOIN administrators a ON i.invited_by = a.id WHERE i.id = $1 AND i.tenant_id = $2', [invitationId, auth.tenantId], logger); if (result.rowCount === 0) throw new NotFoundError('Invitation not found');

const row = result.rows[0]; return success({ invitation: { id: row.id, email: row.email, role: row.role, status: row.status, environment: row.environment, invitedBy: { id: row.invited_by, name: '${row.inviter_first_name}

${row.inviter_last_name}‘ }, message: row.message, expiresAt: row.expires_at, createdAt: row.created_at, acceptedAt: row.accepted_at, }, }); }

async function handleResendInvitation(invitationId: string, event: APIGatewayProxyEvent, logger: Logger): Promise { const auth = extractAuthContext(event); requireAdmin(auth); requirePermission(auth, ‘admin:write’);

const result = await executeQuery(‘SELECT * FROM invitations WHERE id = $1 AND tenant_id = $2‘, [invitationId, auth.tenantId], logger); if (result.rowCount === 0) throw new NotFoundError(‘Invitation not found’);

const invitation = result.rows[0]; if (invitation.status !== ‘pending’) throw new ValidationError(‘Cannot resend ${invitation.status} invitation‘);

const { token, tokenHash } = generateInvitationToken(); const expiresAt = calculateExpiry(48);

await executeQuery(‘UPDATE invitations SET token_hash = $2, expires_at = $3 WHERE id = $1‘, [invitationId, tokenHash, expiresAt], logger);

const inviterResult = await executeQuery(‘SELECT first_name, last_name FROM administrators WHERE id = $1‘, [auth.userId], logger); const inviterName = inviterResult.rows[0] ? ‘${inviterResult.rows[0].first_name} ${inviterResult.rows[0].last_name}‘ : ‘An administrator’;

const appResult = await executeQuery(‘SELECT name FROM apps WHERE id = $1‘, [auth.appId], logger); const appName = appResult.rows[0]?.name || auth.appId;

const acceptUrl = ‘${process.env.ADMIN_URL}/invite/accept?token=${token}‘; const emailContent = generateInvitationEmail({ inviteeName: ’’, inviterName, role: invitation.role, appName, environment: invitation.environment, acceptUrl, expiresAt, message: invitation.message }); await sendEmail({ to: invitation.email, subject: ‘Reminder: You’ve been invited to ${appName}‘, html: emailContent.html, text: emailContent.text }, logger);

logger.info(‘Invitation resent’, { invitationId, email: invitation.email }); return success({ message: ‘Invitation resent successfully’, expiresAt }); }

async function handleRevokeInvitation(invitationId: string, event: APIGatewayProxyEvent, logger: Logger): Promise { const auth = extractAuthContext(event); requireAdmin(auth); requirePermission(auth, ‘admin:write’);

const result = await executeQuery(‘SELECT * FROM invitations WHERE id = $1 AND tenant_id = $2‘, [invitationId, auth.tenantId], logger); if (result.rowCount === 0) throw new NotFoundError(‘Invitation not found’);

const invitation = result.rows[0]; if (invitation.status !== ‘pending’) throw new ValidationError(‘Cannot revoke ${invitation.status} invitation‘);

await executeQuery(‘UPDATE invitations SET status = ‘revoked’ WHERE id = $1‘, [invitationId], logger);

await createAuditLog({ tenant_id: auth.tenantId, user_id: null, admin_id: auth.userId, action: ‘invitation.revoke’, resource_type: ‘invitation’, resource_id: invitationId, details: { email: invitation.email }, ip_address: event.requestContext.identity?.sourceIp || null, user_agent: event.headers[‘User-Agent’] || null, }, logger);

logger.info(‘Invitation revoked’, { invitationId, email: invitation.email }); return success({ message: ‘Invitation revoked successfully’ }); } “‘

PART 3: TWO-PERSON APPROVALS LAMBDA

packages/infrastructure/lambda/admin/approvals.ts

“‘typescript /** * Two-Person Approval Workflow Lambda * Production deployments require separate initiator and approver */

import { APIGatewayProxyEvent, APIGatewayProxyResult, Context } from ‘aws-lambda’; import { v4 as uuidv4 } from ‘uuid’; import { Logger } from ‘../shared/logger’; import { success, created, handleError } from ‘../shared/response’; import { extractAuthContext, requireAdmin, requirePermission } from ‘../shared/auth’; import { ValidationError, NotFoundError, ForbiddenError } from ‘../shared/errors’; import { executeQuery, createAuditLog } from ‘../shared/db’; import { createApprovalSchema, processApprovalSchema, ApprovalStatus, AdminRole, ROLE_HIERARCHY } from ‘../shared/admin/types’; import { calculateExpiry, isExpired } from ‘../shared/admin/tokens’; import { sendEmail, generateApprovalEmail } from ‘../shared/admin/email’;

const logger = new Logger({ handler: ‘approvals’ });

export async function handler(event: APIGatewayProxyEvent, context: Context): Promise { const requestLogger = logger.child({ requestId: context.awsRequestId, path: event.path });

try { const auth = extractAuthContext(event); requireAdmin(auth);

```
const approvalId = event.pathParameters?.approvalId;
const action = event.path.split('/').pop();

switch (event.httpMethod) {
  case 'GET':
    if (approvalId) return await handleGetApproval(approvalId, auth, requestLogger);
    return await handleListApprovals(event, auth, requestLogger);
  case 'POST':
    if (action === 'process' && approvalId) return await handleProcessApproval(approvalId, e
    return await handleCreateApproval(event, auth, requestLogger);
  case 'DELETE':
    if (!approvalId) throw new ValidationError('Approval ID required');
```

```
    return await handleCancelApproval(approvalId, event, auth, requestLogger);
  default:
    throw new ValidationError(\`Method \${event.httpMethod} not allowed\`);
}
```

} catch (error) { return handleError(error, requestLogger); } }

async function handleCreateApproval(event: APIGatewayProxyEvent, auth: ReturnType, logger: Logger): Promise { requirePermission(auth, 'approvals:initiate');

const body = event.body ? JSON.parse(event.body) : {}; const parseResult = createApprovalSchema.safeParse(body); if (!parseResult.success) throw new ValidationError('Invalid request body', parseResult.error.flatten().fieldErrors as Record<string, string[]>);

const { type, action, resourceType, resourceId, details, priority, notes, expiresInHours } = parseResult.data; const requiresTwoPersonApproval = auth.environment === 'prod';

const requesterResult = await executeQuery('SELECT first_name, last_name FROM administrators WHERE id = $1', [auth.userId], logger); const requesterName = requesterResult.rows[0] ? `${requesterResult.rows[0].first_name} ${requesterResult.rows[0].last_name}` : 'Unknown';

const approvalId = uuidv4(); const expiresAt = calculateExpiry(expiresInHours);

await executeQuery('INSERT INTO approval_requests (id, type, app_id, tenant_id, environment, requested_by, requested_at, expires_at, status, action, resource_type, resource_id, details, priority, notes, requires_two_person, created_at) VALUES ($1, $2, $3, $4, $5, $6, NOW(), $7, 'pending', $8, $9, $10, $11, $12, $13, $14, NOW())', [approvalId, type, auth.appId, auth.tenantId, auth.environment, auth.userId, expiresAt, action, resourceType, resourceId, JSON.stringify(details), priority, notes || null, requiresTwoPersonApproval], logger);

if (requiresTwoPersonApproval) { await notifyApprovers(approvalId, auth, requesterName, logger); }

await createAuditLog({ tenant_id: auth.tenantId, user_id: null, admin_id: auth.userId, action: 'approval.create', resource_type: 'approval_request', resource_id: approvalId, details: { type, resourceType, resourceId, requiresTwoPersonApproval }, ip_address: event.requestContext.identity?.sourceIp || null, user_agent: event.headers['User-Agent'] || null, }, logger);

logger.info('Approval request created', { approvalId, type, requestedBy: auth.userId, requiresTwoPersonApproval }); return created({ approval: { id: approvalId, type, status: 'pending', action, resourceType, resourceId, priority, expiresAt, requiresTwoPersonApproval, createdAt: new Date().toISOString() }, }); }

async function handleProcessApproval(approvalId: string, event: APIGatewayProxyEvent, auth: ReturnType, logger: Logger): Promise { requirePermission(auth, 'approvals:*'); const body = event.body ? JSON.parse(event.body) : {}; const parseResult = processApprovalSchema.safeParse(body); if (!parseResult.success) throw new ValidationError('Invalid request body', parseResult.error.flatten().fieldErrors as Record<string, string[]>); const { action, reason } = parseResult.data; const result = await executeQuery('SELECT * FROM approval_requests WHERE id = $1 AND tenant_id = $2', [approvalId, auth.tenantId], logger); if (result.rowCount === 0) throw new NotFoundError('Approval request not found'); const approval = result.rows[0]; if (approval.status !== 'pending') throw new ValidationError(`Cannot process ${approval.status} approval request`); if (isExpired(approval.expires_at)) { await executeQuery('UPDATE approval_requests SET status = 'expired' WHERE id = $1', [approvalId], logger); throw new ValidationError('This approval request has expired'); } // Two-person approval: cannot approve own request if (approval.requires_two_person && approval.requested_by === auth.userId) { throw new ForbiddenError('You cannot approve your own request. Production deployments require approval from a different administrator.'); } const newStatus: ApprovalStatus = action === 'approve' ? 'approved' : 'rejected'; await executeQuery(`UPDATE approval_requests SET status = $2, approved_by = $3, approved_at = NOW(), rejected_reason = $4 WHERE id = $1`, [approvalId, newStatus, auth.userId, action === 'reject' ? reason : null], logger); if (action === 'approve') { await executeApprovedAction(approval, logger); } await createAuditLog({ tenant_id: auth.tenantId, user_id: null, admin_id: auth.userId, action: `approval.${action}`, resource_type: 'approval_request', resource_id: approvalId, details: { type: approval.type, resourceType: approval.resource_type, resourceId: approval.resource_id, reason }, ip_address: event.requestContext.identity?.sourceIp || null, user_agent: event.headers['User-Agent'] || null, }, logger); logger.info('Approval request processed', { approvalId, action, processedBy: auth.userId }); return success({ approval: { id: approvalId, status: newStatus, processedBy: auth.userId, processedAt: new Date().toISOString(), reason: action === 'reject' ? reason : undefined } }); } async function executeApprovedAction(approval: any, logger: Logger): Promise { logger.info('Executing approved action', { type: approval.type, resourceType: approval.resource_type, resourceId: approval.resource_id });

switch (approval.type) { case 'deployment': // Trigger deployment workflow via Step Functions or CodePipeline break; case 'promotion': await executeQuery('UPDATE deployments SET promoted_to = $2, promoted_at = NOW() WHERE id = $1', [approval.resource_id, approval.details?.targetEnv], logger); break; case 'model_activation': await executeQuery('UPDATE ai_models SET status = $2, thermal_state = $3, updated_at = NOW() WHERE id = $1', [approval.details?.modelId, approval.details?.newStatus, approval.details?.thermalState], logger); break; case 'provider_change': await executeQuery('UPDATE ai_providers SET config = $2, updated_at = NOW() WHERE id = $1', [approval.details?.providerId, JSON.stringify(approval.details?.config)], logger); break; case 'user_role_change': await executeQuery('UPDATE administrators SET role = $2, updated_at = NOW() WHERE id = $1', [approval.details?.userId, approval.details?.newRole], logger); break; case 'billing_change': await executeQuery('UPDATE billing_settings SET margin_percent = COALESCE($2, margin_percent), tax_percent = COALESCE($3, tax_percent), updated_at = NOW() WHERE tenant_id = $1', [approval.details?.tenantId, approval.details?.settings?.marginPercent, approval.details?.settings?.taxPercent], logger); break; default: logger.warn('Unknown approval type', { type: approval.type }); } }

async function notifyApprovers(approvalId: string, auth: ReturnType, requesterName: string, logger: Logger): Promise { const result = await executeQuery('SELECT id, email, first_name, last_name FROM administrators WHERE tenant_id = $1 AND id != $2 AND status = 'active' AND role IN ('super_admin', 'admin')', [auth.tenantId, auth.userId], logger); if (result.rowCount === 0) { logger.warn('No other admins available to approve'); return; }

const appResult = await executeQuery('SELECT name FROM apps WHERE id = $1', [auth.appId], logger); const appName = appResult.rows[0]?.name || auth.appId;

const approvalResult = await executeQuery('SELECT * FROM approval_requests WHERE id = $1', [approvalId], logger); const approval = approvalResult.rows[0];

for (const admin of result.rows) { const approveUrl = '${process.env.ADMIN_URL}/approvals/${approvalId}? const rejectUrl = '${process.env.ADMIN_URL}/approvals/${approvalId}?action=reject'; const emailContent = generateApprovalEmail({ approverName: admin.first_name, requesterName, appName, environment: auth.environment, action: approval.action, resourceType: approval.resource_type, resourceId: approval.resource_id, approveUrl, rejectUrl, expiresAt: approval.expires_at, }); await sendEmail({ to: admin.email, subject: 'Ã¢Å¡Â Ã¯Â¸Â Approval Required: ${approval.action} - ${appName}', html: emailContent.html, text: emailContent.text }, logger); } logger.info('Approvers notified', { approvalId, notifiedCount: result.rowCount }); }

async function handleListApprovals(event: APIGatewayProxyEvent, auth:

ReturnType, logger: Logger): Promise { requirePermission(auth, ‘approvals:read’);

const status = event.queryStringParameters?.status; const pendingForMe = event.queryStringParameters?.pendingForMe === ‘true’; const limit = parseInt(event.queryStringParameters?.limit || ‘50’); const offset = parseInt(event.queryStringParameters?.offset || ‘0’);

let query = ‘SELECT ar.*, req.first_name as requester_first_name, req.last_name as requester_last_name FROM approval_requests ar LEFT JOIN administrators req ON ar.requested_by = req.id WHERE ar.tenant_id = $1‘; const params: any[] = [auth.tenantId];

if (status) { params.push(status); query += ‘ AND ar.status = $${params.length}‘; } if (pendingForMe) { params.push(auth.userId); query += ‘ AND ar.status = ‘pending’ AND ar.requested_by != $${params.length}‘; } query += ‘ ORDER BY ar.created_at DESC LIMIT $${params.length + 1} OFFSET $${params.length + 2}‘; params.push(limit, offset);

const result = await executeQuery(query, params, logger); const approvals = result.rows.map(row => ({ id: row.id, type: row.type, status: row.status, environment: row.environment, action: row.action, resourceType: row.resource_type, resourceId: row.resource_id, priority: row.priority, requestedBy: { id: row.requested_by, name: ‘${row.requester_first_name} ${row.requester_last_name}‘ }, requestedAt: row.requested_at, expiresAt: row.expires_at, requiresTwoPersonApproval: row.requires_two_person, canApprove: row.status === ‘pending’ && row.requested_by !== auth.userId, }));

return success({ approvals, pagination: { limit, offset, hasMore: approvals.length === limit } }); }

async function handleGetApproval(approvalId: string, auth: ReturnType, logger: Logger): Promise { requirePermission(auth, ‘approvals:read’);

const result = await executeQuery(‘SELECT ar.*, req.first_name as requester_first_name, req.last_name as requester_last_name, req.email as requester_email FROM approval_requests ar LEFT JOIN administrators req ON ar.requested_by = req.id WHERE ar.id = $1 AND ar.tenant_id = $2‘, [approvalId, auth.tenantId], logger); if (result.rowCount === 0) throw new NotFoundError(‘Approval request not found’);

const row = result.rows[0]; return success({ approval: { id: row.id, type: row.type, status: row.status, environment: row.environment, action: row.action, resourceType: row.resource_type, resourceId: row.resource_id, details: row.details, priority: row.priority, notes: row.notes, requestedBy: { id: row.requested_by, name: ‘${row.requester_first_name} ${row.requester_last_name}‘, email: row.requester_email }, requestedAt: row.requested_at, expiresAt: row.expires_at, requiresTwoPersonApproval:

row.requires_two_person, canApprove: row.status === 'pending' && row.requested_by !== auth.userId, }, }); }

async function handleCancelApproval(approvalId: string, event: APIGatewayProxyEvent, auth: ReturnType, logger: Logger): Promise { const result = await executeQuery(`SELECT * FROM approval_requests WHERE id = $1 AND tenant_id = $2`, [approvalId, auth.tenantId], logger); if (result.rowCount === 0) throw new NotFoundError('Approval request not found');

const approval = result.rows[0]; if (approval.requested_by !== auth.userId && auth.role !== AdminRole.SUPER_ADMIN) { throw new ForbiddenError('Only the requester or a super admin can cancel this request'); } if (approval.status !== 'pending') throw new ValidationError(`Cannot cancel ${approval.status} approval request`);

await executeQuery(`UPDATE approval_requests SET status = 'cancelled' WHERE id = $1`, [approvalId], logger); await createAuditLog({ tenant_id: auth.tenantId, user_id: null, admin_id: auth.userId, action: 'approval.cancel', resource_type: 'approval_request', resource_id: approvalId, details: { type: approval.type }, ip_address: event.requestContext.identity?.sourceIp || null, user_agent: event.headers['User-Agent'] || null, }, logger);

logger.info('Approval request cancelled', { approvalId }); return success({ message: 'Approval request cancelled' }); } ```

PART 4: METERING LAMBDA

packages/infrastructure/lambda/billing/metering.ts

```typescript /** * Usage Metering Lambda * Collects and stores usage events for billing calculations */

import { APIGatewayProxyEvent, APIGatewayProxyResult, Context } from 'aws-lambda'; import { DynamoDBClient } from '@aws-sdk/client-dynamodb'; import { DynamoDBDocumentClient, PutCommand, QueryCommand, UpdateCommand } from '@aws-sdk/lib-dynamodb'; import { v4 as uuidv4 } from 'uuid'; import { z } from 'zod'; import { Logger } from '../shared/logger'; import { success, created, handleError } from '../shared/response'; import { extractAuthContext } from '../shared/auth'; import { ValidationError } from '../shared/errors'; import { executeQuery } from '../shared/db';

const logger = new Logger({ handler: 'metering' }); const ddbClient = DynamoDBDocumentClient.from(new DynamoDBClient({}), { marshallOptions: { removeUndefinedValues: true } });

const USAGE_TABLE = process.env.USAGE_TABLE || 'radiant-usage-events'; const ROLLUP_TABLE = process.env.ROLLUP_TABLE || 'radiant-usage-rollups';
```

const recordUsageSchema = z.object({ requestId: z.string(), providerId: z.string(), modelId: z.string(), modelName: z.string(), requestType: z.enum([‘chat’, ‘embedding’, ‘image’, ‘audio’, ‘video’]), inputTokens: z.number().int().min(0), outputTokens: z.number().int().min(0), latencyMs: z.number().int().min(0), cached: z.boolean().default(false), phiDetected: z.boolean().default(false), phiSanitized: z.boolean().default(false), userId: z.string().optional(), });

export async function handler(event: APIGatewayProxyEvent, context: Context): Promise { const requestLogger = logger.child({ requestId: context.awsRequestId, path: event.path });

try { const auth = extractAuthContext(event); const action = event.path.split(‘/’).pop();

```
switch (event.httpMethod) {
  case 'POST':
    if (action === 'record') return await handleRecordUsage(event, auth, requestLogger);
    if (action === 'batch') return await handleBatchRecord(event, auth, requestLogger);
    break;
  case 'GET':
    if (action === 'summary') return await handleGetSummary(event, auth, requestLogger);
    if (action === 'rollups') return await handleGetRollups(event, auth, requestLogger);
    break;
}
throw new ValidationError(\`Unknown action: \${action}\`);
```

} catch (error) { return handleError(error, requestLogger); } }

async function handleRecordUsage(event: APIGatewayProxyEvent, auth: ReturnType, logger: Logger): Promise { const body = event.body ? JSON.parse(event.body) : {}; const parseResult = recordUsageSchema.safeParse(body); if (!parseResult.success) throw new ValidationError(‘Invalid usage data’, parseResult.error.flatten().fieldErrors as Record<string, string[]>);

const data = parseResult.data;

// Get pricing from model registry const pricing = await getModelPricing(data.modelId, logger); const inputCost = (data.inputTokens / 1000000) * pricing.inputPricePerMillion; const outputCost = (data.outputTokens / 1000000) * pricing.outputPricePerMillion; const providerCost = inputCost + outputCost;

// Apply margin const marginPercent = await getTenantMargin(auth.tenantId, logger); const billedCost = providerCost * (1 + marginPercent / 100);

const usageEvent = { id: uuidv4(), timestamp: new Date().toISOString(), tenantId: auth.tenantId, userId: data.userId || auth.userId, adminId: auth.isAdmin ? auth.userId : undefined, appId: auth.appId, environment: auth.environment, providerId: data.providerId, modelId: data.modelId,

modelName: data.modelName, requestType: data.requestType, inputTokens: data.inputTokens, outputTokens: data.outputTokens, totalTokens: data.inputTokens + data.outputTokens, providerCost, billedCost, currency: ‘USD’, requestId: data.requestId, latencyMs: data.latencyMs, cached: data.cached, phiDetected: data.phiDetected, phiSanitized: data.phiSanitized, };

await ddbClient.send(new PutCommand({ TableName: USAGE_TABLE, Item: { pk: ‘TENANT#${auth.tenantId}‘, sk: ‘EVENT#${usageEvent.timestamp}#${usageEvent.id}‘, …usageEvent, ttl: Math.floor(Date.now() / 1000) + (90 * 24 * 60 * 60) }, }));

await updateDailyRollup(usageEvent, logger);

logger.info(‘Usage recorded’, { eventId: usageEvent.id, modelId: data.modelId, tokens: usageEvent.totalTokens, cost: billedCost }); return created({ event: { id: usageEvent.id, billedCost, providerCost } }); }

async function handleBatchRecord(event: APIGatewayProxyEvent, auth: ReturnType, logger: Logger): Promise { const body = event.body ? JSON.parse(event.body) : {}; if (!Array.isArray(body.events) || body.events.length === 0) throw new ValidationError(‘events array is required’); if (body.events.length > 100) throw new ValidationError(‘Maximum 100 events per batch’);

const results: Array<{ id: string; success: boolean; error?: string }> = [];

for (const eventData of body.events) { try { const parseResult = recordUsageSchema.safeParse(eventData); if (!parseResult.success) { results.push({ id: eventData.requestId || ‘unknown’, success: false, error: ‘Invalid data’ }); continue; }

```
  const data = parseResult.data;
  const pricing = await getModelPricing(data.modelId, logger);
  const providerCost = ((data.inputTokens / 1000000) * pricing.inputPricePerMillion) + ((dat
  const marginPercent = await getTenantMargin(auth.tenantId, logger);
  const billedCost = providerCost * (1 + marginPercent / 100);

  const usageEvent = {
    id: uuidv4(), timestamp: new Date().toISOString(), tenantId: auth.tenantId, userId: data
    appId: auth.appId, environment: auth.environment, providerId: data.providerId, modelId:
    requestType: data.requestType, inputTokens: data.inputTokens, outputTokens: data.outputT
    providerCost, billedCost, currency: 'USD', requestId: data.requestId, latencyMs: data.la
  };

  await ddbClient.send(new PutCommand({ TableName: USAGE_TABLE, Item: { pk: \`TENANT#\${auth
  await updateDailyRollup(usageEvent, logger);
  results.push({ id: usageEvent.id, success: true });
} catch (error: any) {
  results.push({ id: eventData.requestId || 'unknown', success: false, error: error.message
}
```

}

logger.info('Batch usage recorded', { total: body.events.length, successful: results.filter(r => r.success).length }); return success({ results, summary: { total: results.length, successful: results.filter(r => r.success).length, failed: results.filter(r => !r.success).length } }); }

async function handleGetSummary(event: APIGatewayProxyEvent, auth: ReturnType, logger: Logger): Promise { const startDate = event.queryStringParameters?.startDate || getDefaultStartDate(); const endDate = event.queryStringParameters?.endDate || getTodayDate();

const response = await ddbClient.send(new QueryCommand({ TableName: ROLLUP_TABLE, KeyConditionExpression: 'pk = :pk AND sk BETWEEN :start AND :end', ExpressionAttributeValues: { ':pk': `TENANT#${auth.tenantId}`, ':start': `DATE#${startDate}`, ':end': `DATE#${endDate}#~` }, }));

const rollups = response.Items || []; const totals = rollups.reduce((acc, r) => ({ requests: acc.requests + (r.requestCount || 0), inputTokens: acc.inputTokens + (r.inputTokens || 0), outputTokens: acc.outputTokens + (r.outputTokens || 0), providerCost: acc.providerCost + (r.providerCost || 0), billedCost: acc.billedCost + (r.billedCost || 0), }), { requests: 0, inputTokens: 0, outputTokens: 0, providerCost: 0, billedCost: 0 });

return success({ period: { startDate, endDate }, totals }); }

async function handleGetRollups(event: APIGatewayProxyEvent, auth: ReturnType, logger: Logger): Promise { const startDate = event.queryStringParameters?.startDate || getDefaultStartDate(); const endDate = event.queryStringParameters?.endDate || getTodayDate();

const response = await ddbClient.send(new QueryCommand({ TableName: ROLLUP_TABLE, KeyConditionExpression: 'pk = :pk AND sk BETWEEN :start AND :end', ExpressionAttributeValues: { ':pk': `TENANT#${auth.tenantId}`, ':start': `DATE#${startDate}`, ':end': `DATE#${endDate}#~` }, }));

const rollups = (response.Items || []).map(item => ({ date: item.date, modelId: item.modelId, providerId: item.providerId, requestCount: item.requestCount, inputTokens: item.inputTokens, outputTokens: item.outputTokens, totalTokens: item.totalTokens, providerCost: item.providerCost, billedCost: item.billedCost, avgLatencyMs: item.avgLatencyMs, }));

return success({ rollups, period: { startDate, endDate } }); }

async function updateDailyRollup(event: any, logger: Logger): Promise { const date = event.timestamp.split('T')[0]; try { await ddbClient.send(new UpdateCommand({ TableName: ROLLUP_TABLE, Key: { pk: `TENANT#${event.tenantId}`, sk: `DATE#${date}#MODEL#${event.modelId}` }, UpdateExpression: 'SET #date = :date, modelId = :modelId, providerId

= :providerId, requestCount = if_not_exists(requestCount, :zero) + :one, inputTokens = if_not_exists(inputTokens, :zero) + :inputTokens, outputTokens = if_not_exists(outputTokens, :zero) + :outputTokens, totalTokens = if_not_exists(totalTokens, :zero) + :totalTokens, providerCost = if_not_exists(providerCost, :zero) + :providerCost, billedCost = if_not_exists(billedCost, :zero) + :billedCost, cachedRequests = if_not_exists(cachedRequests, :zero) + :cached, phiRequests = if_not_exists(phiRequests, :zero) + :phi, updatedAt = :updatedAt‘, ExpressionAttributeNames: { ‘#date’: ‘date’ }, ExpressionAttributeValues: { ‘:date’: date, ‘:modelId’: event.modelId, ‘:providerId’: event.providerId, ‘:zero’: 0, ‘:one’: 1, ‘:inputTokens’: event.inputTokens, ‘:outputTokens’: event.outputTokens, ‘:totalTokens’: event.totalTokens, ‘:providerCost’: event.providerCost, ‘:billedCost’: event.billedCost, ‘:cached’: event.cached ? 1 : 0, ‘:phi’: event.phiDetected ? 1 : 0, ‘:updatedAt’: new Date().toISOString(), }, })); } catch (error) { logger.error(‘Failed to update rollup’, error as Error, { tenantId: event.tenantId, date, modelId: event.modelId }); } }

async function getModelPricing(modelId: string, logger: Logger): Promise<{ inputPricePerMillion: number; outputPricePerMillion: number }> { const result = await executeQuery(‘SELECT input_price_per_million, output_price_per_million FROM ai_models WHERE id = $1‘, [modelId], logger); if (result.rowCount === 0) return { inputPricePerMillion: 1.0, outputPricePerMillion: 2.0 }; return { inputPricePerMillion: parseFloat(result.rows[0].input_price_per_million) || 1.0, outputPricePerMillion: parseFloat(result.rows[0].output_price_per_million) || 2.0 }; }

async function getTenantMargin(tenantId: string, logger: Logger): Promise { const result = await executeQuery(‘SELECT margin_percent FROM billing_settings WHERE tenant_id = $1‘, [tenantId], logger); return result.rowCount > 0 ? parseFloat(result.rows[0].margin_percent) : 20; }

function getDefaultStartDate(): string { const d = new Date(); d.setDate(d.getDate() - 30); return d.toISOString().split(‘T’)[0]; } function getTodayDate(): string { return new Date().toISOString().split(‘T’)[0]; } “‘

PART 5: DATABASE SCHEMA ADDITIONS

packages/infrastructure/migrations/005_admin_billing.sql

“‘sql – == – RADIANT v2.2.0 - Admin & Billing Schema – ==

– Admin Invitations CREATE TABLE IF NOT EXISTS invitations (id UUID PRIMARY KEY DEFAULT gen_random_uuid(), email VARCHAR(255) NOT NULL, role VARCHAR(50) NOT NULL, invited_by UUID NOT NULL REFERENCES administrators(id), app_id VARCHAR(100) NOT NULL, tenant_id VARCHAR(100) NOT NULL, environment VARCHAR(20)

NOT NULL, token_hash VARCHAR(64) NOT NULL, expires_at TIMESTAMP WITH TIME ZONE NOT NULL, status VARCHAR(20) NOT NULL DEFAULT ‘pending’, message TEXT, accepted_at TIMESTAMP WITH TIME ZONE, accepted_by_ip VARCHAR(45), created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(), CONSTRAINT valid_invitation_status CHECK (status IN (‘pending’, ‘accepted’, ‘expired’, ‘revoked’)));

CREATE INDEX idx_invitations_email ON invitations(email); CREATE INDEX idx_invitations_tenant ON invitations(tenant_id); CREATE INDEX idx_invitations_token ON invitations(token_hash); CREATE INDEX idx_invitations_status ON invitations(status);

– Approval Requests CREATE TABLE IF NOT EXISTS approval_requests (id UUID PRIMARY KEY DEFAULT gen_random_uuid(), type VARCHAR(50) NOT NULL, app_id VARCHAR(100) NOT NULL, tenant_id VARCHAR(100) NOT NULL, environment VARCHAR(20) NOT NULL, requested_by UUID NOT NULL REFERENCES administrators(id), requested_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(), expires_at TIMESTAMP WITH TIME ZONE NOT NULL, status VARCHAR(20) NOT NULL DEFAULT ‘pending’, approved_by UUID REFERENCES administrators(id), approved_at TIMESTAMP WITH TIME ZONE, rejected_reason TEXT, action VARCHAR(100) NOT NULL, resource_type VARCHAR(100) NOT NULL, resource_id VARCHAR(255), details JSONB, priority VARCHAR(20) NOT NULL DEFAULT ‘medium’, notes TEXT, requires_two_person BOOLEAN NOT NULL DEFAULT false, created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(), CONSTRAINT valid_approval_status CHECK (status IN (‘pending’, ‘approved’, ‘rejected’, ‘expired’, ‘cancelled’)), CONSTRAINT valid_priority CHECK (priority IN (‘low’, ‘medium’, ‘high’, ‘critical’)));

CREATE INDEX idx_approvals_tenant ON approval_requests(tenant_id); CREATE INDEX idx_approvals_status ON approval_requests(status); CREATE INDEX idx_approvals_requested_by ON approval_requests(requested_by);

– Admin Profiles CREATE TABLE IF NOT EXISTS admin_profiles (admin_id UUID PRIMARY KEY REFERENCES administrators(id) ON DELETE CASCADE, notifications JSONB NOT NULL DEFAULT ‘{}’, timezone VARCHAR(50) NOT NULL DEFAULT ‘America/New_York’, language VARCHAR(10) NOT NULL DEFAULT ‘en’, date_format VARCHAR(20) NOT NULL DEFAULT ‘MM/DD/YYYY’, time_format VARCHAR(10) NOT NULL DEFAULT ‘12h’, currency VARCHAR(3) NOT NULL DEFAULT ‘USD’, theme VARCHAR(20) NOT NULL DEFAULT ‘system’, default_environment VARCHAR(20) NOT NULL DEFAULT ‘dev’, sidebar_collapsed BOOLEAN NOT NULL DEFAULT false, table_rows_per_page INTEGER NOT NULL DEFAULT 25, updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW());

– Billing Settings CREATE TABLE IF NOT EXISTS billing_settings (tenant_id VARCHAR(100) PRIMARY KEY, margin_percent DECIMAL(5,2) NOT NULL DEFAULT 20.00, margin_type VARCHAR(20) NOT NULL DEFAULT ‘fixed’, tiers JSONB, tax_enabled BOOLEAN NOT NULL DEFAULT false, tax_percent DECIMAL(5,2) NOT NULL DEFAULT 0.00, tax_id VARCHAR(50), stripe_customer_id VARCHAR(100), default_payment_method_id VARCHAR(100), auto_pay BOOLEAN NOT NULL DEFAULT false, billing_cycle_day INTEGER NOT NULL DEFAULT 1, currency VARCHAR(3) NOT NULL DEFAULT ‘USD’, budget_limit DECIMAL(12,2), alert_thresholds INTEGER[] NOT NULL DEFAULT ‘{50, 75, 90, 100}’, created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(), updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(), CONSTRAINT valid_margin_type CHECK (margin_type IN (‘fixed’, ‘tiered’)), CONSTRAINT valid_billing_day CHECK (billing_cycle_day BETWEEN 1 AND 28));

– Invoices CREATE TABLE IF NOT EXISTS invoices (id UUID PRIMARY KEY DEFAULT gen_random_uuid(), tenant_id VARCHAR(100) NOT NULL, app_id VARCHAR(100) NOT NULL, period_start DATE NOT NULL, period_end DATE NOT NULL, subtotal DECIMAL(12,2) NOT NULL, margin_percent DECIMAL(5,2) NOT NULL, tax DECIMAL(12,2) NOT NULL DEFAULT 0, tax_percent DECIMAL(5,2) NOT NULL DEFAULT 0, total DECIMAL(12,2) NOT NULL, currency VARCHAR(3) NOT NULL DEFAULT ‘USD’, status VARCHAR(20) NOT NULL DEFAULT ‘draft’, due_date TIMESTAMP WITH TIME ZONE NOT NULL, paid_at TIMESTAMP WITH TIME ZONE, line_items JSONB NOT NULL DEFAULT ‘[]’, stripe_invoice_id VARCHAR(100), stripe_payment_intent_id VARCHAR(100), created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(), updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(), CONSTRAINT valid_invoice_status CHECK (status IN (‘draft’, ‘pending’, ‘paid’, ‘overdue’, ‘cancelled’)));

CREATE INDEX idx_invoices_tenant ON invoices(tenant_id); CREATE INDEX idx_invoices_status ON invoices(status); CREATE INDEX idx_invoices_period ON invoices(period_start, period_end);

– Admin Notifications CREATE TABLE IF NOT EXISTS admin_notifications (id UUID PRIMARY KEY DEFAULT gen_random_uuid(), admin_id UUID NOT NULL REFERENCES administrators(id) ON DELETE CASCADE, tenant_id VARCHAR(100) NOT NULL, type VARCHAR(50) NOT NULL, priority VARCHAR(20) NOT NULL DEFAULT ‘medium’, title VARCHAR(255) NOT NULL, message TEXT NOT NULL, action_url VARCHAR(500), action_label VARCHAR(100), read BOOLEAN NOT NULL DEFAULT false, read_at TIMESTAMP WITH TIME ZONE, dismissed BOOLEAN NOT NULL DEFAULT false, dismissed_at TIMESTAMP WITH TIME ZONE, email_sent BOOLEAN NOT NULL DEFAULT false, email_sent_at TIMESTAMP WITH TIME ZONE, created_at TIMESTAMP WITH TIME

ZONE NOT NULL DEFAULT NOW(), expires_at TIMESTAMP WITH TIME ZONE, CONSTRAINT valid_notification_type CHECK (type IN ('security', 'billing', 'deployment', 'approval', 'system', 'alert')), CONSTRAINT valid_notification_priority CHECK (priority IN ('low', 'medium', 'high', 'critical')));

CREATE INDEX idx_notifications_admin ON admin_notifications(admin_id); CREATE INDEX idx_notifications_read ON admin_notifications(admin_id, read); "'

API ROUTES SUMMARY

Admin Routes

Method	Path	Description	Permission
POST	/admin/invitations	Create invitation	admin:write
GET	/admin/invitations	List invitations	admin:read
GET	/admin/invitations/:id	Get invitation	admin:read
POST	/admin/invitations/:id/resend	Resend invitation	admin:write
DELETE	/admin/invitations/:id	Revoke invitation	admin:write
POST	/admin/invitations/accept	Accept invitation	(public)
POST	/admin/approvals	Create approval request	approvals:initiate
GET	/admin/approvals	List approvals	approvals:read
GET	/admin/approvals/:id	Get approval	approvals:read
POST	/admin/approvals/:id/process	Approve/reject	approvals:*
DELETE	/admin/approvals/:id	Cancel approval	approvals:initiate
GET	/admin/users	List admins	admin:read
GET	/admin/users/:id	Get admin	admin:read
PUT	/admin/users/:id	Update admin	admin:write
DELETE	/admin/users/:id	Delete admin	admin:*

Billing Routes

Method	Path	Description	Permission
GET	/billing/settings	Get settings	billing:read
PUT	/billing/settings	Update settings	billing:*
GET	/billing/current	Current period usage	billing:read
GET	/billing/projections	Cost projections	billing:read
GET	/billing/invoices	List invoices	billing:read
GET	/billing/invoices/:id	Get invoice	billing:read
POST	/billing/generate	Generate invoice	billing:*

Metering Routes

Method	Path	Description	Permission
POST	/metering/record	Record usage event	(internal)
POST	/metering/batch	Batch record events	(internal)
GET	/metering/summary	Usage summary	billing:read
GET	/metering/rollups	Daily rollups	billing:read

DEPLOYMENT VERIFICATION

“‘bash # 1. Create Invitation curl -X POST -H “Authorization: Bearer $ADMIN_TOKEN” -H “Content-Type: application/json” \ -d ‘{“email”: “new.admin@company.com”, “role”: “operator”, “message”: “Welcome!”, “expiresInHours”: 48}’ \ https://admin-api.thinktank.YOUR_DOMAIN.com/api/v2/admin/invitations

2. List Pending Approvals (for me to approve)

curl -H “Authorization: Bearer $ADMIN_TOKEN” \ “https://admin-api.thinktank.YOUR_DOMAIN.com/api/v2/admin/approvals?status=pending&pendingForMe=true”

3. Get Current Billing Period

curl -H “Authorization: Bearer $ADMIN_TOKEN” \ https://admin-api.thinktank.YOUR_DOMAIN.com/api/v2/billing/current

4. Get Usage Rollups

curl -H “Authorization: Bearer $ADMIN_TOKEN” \ “https://admin-api.thinktank.YOUR_DOMAIN.com/api/v2/metering/rollups?startDate=2024-12-01&endDate=2024-12-21”

5. Generate Invoice

curl -X POST -H “Authorization: Bearer $ADMIN_TOKEN” -H “Content-Type: application/json” \ -d ‘{“periodStart”: “2024-12-01”, “periodEnd”: “2024-12-31”, “sendToStripe”: true}’ \ https://admin-api.thinktank.YOUR_DOMAIN.com/api/v2/billing/ge
“‘

ESTIMATED COSTS BY TIER

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Lambda (Admin)	$5	$15	$50	$150	$500
Lambda (Billing)	$5	$20	$75	$200	$600
DynamoDB (Usage)	$10	$50	$200	$500	$1,500
SES (Emails)	$1	$5	$20	$50	$150
Stripe Fees	Variable	Variable	Variable	Variable	Variable
Prompt 5 Total	~$21	~$90	~$345	~$900	~$2,750

Note: Stripe fees are 2.9% + $0.30 per transaction, not included in estimates.

NEXT PROMPTS

Continue with: - **Prompt 6**: Self-Hosted Models & Mid-Level Services Configuration - **Prompt 7**: External Providers & Database Schema/Migrations - **Prompt 8**: Admin Web Dashboard (Next.js) - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 5: Lambda Functions - Admin & Billing RADIANT v2.2.0 - December 2024

â• â•

END OF SECTION 5

â• â•

â• â•

SECTION-06-SELF-HOSTED-MODELS

SECTION 6: SELF-HOSTED MODELS & MID-LEVEL SERVICES (v2.2.0)

â•

> **Dependencies:** Sections 0-5 **Creates:** 30+ SageMaker models, Thermal state management

Type Imports

```
import { ThermalState, ThermalConfig, ServiceState, MidLevelService } from '@radiant/shared'
```

RADIANT v2.2.0 - Prompt 6: Self-Hosted Models & Mid-Level Services

> **Prompt 6 of 9** | Target Size: ~75KB | Version: 3.7.0 | December 2024

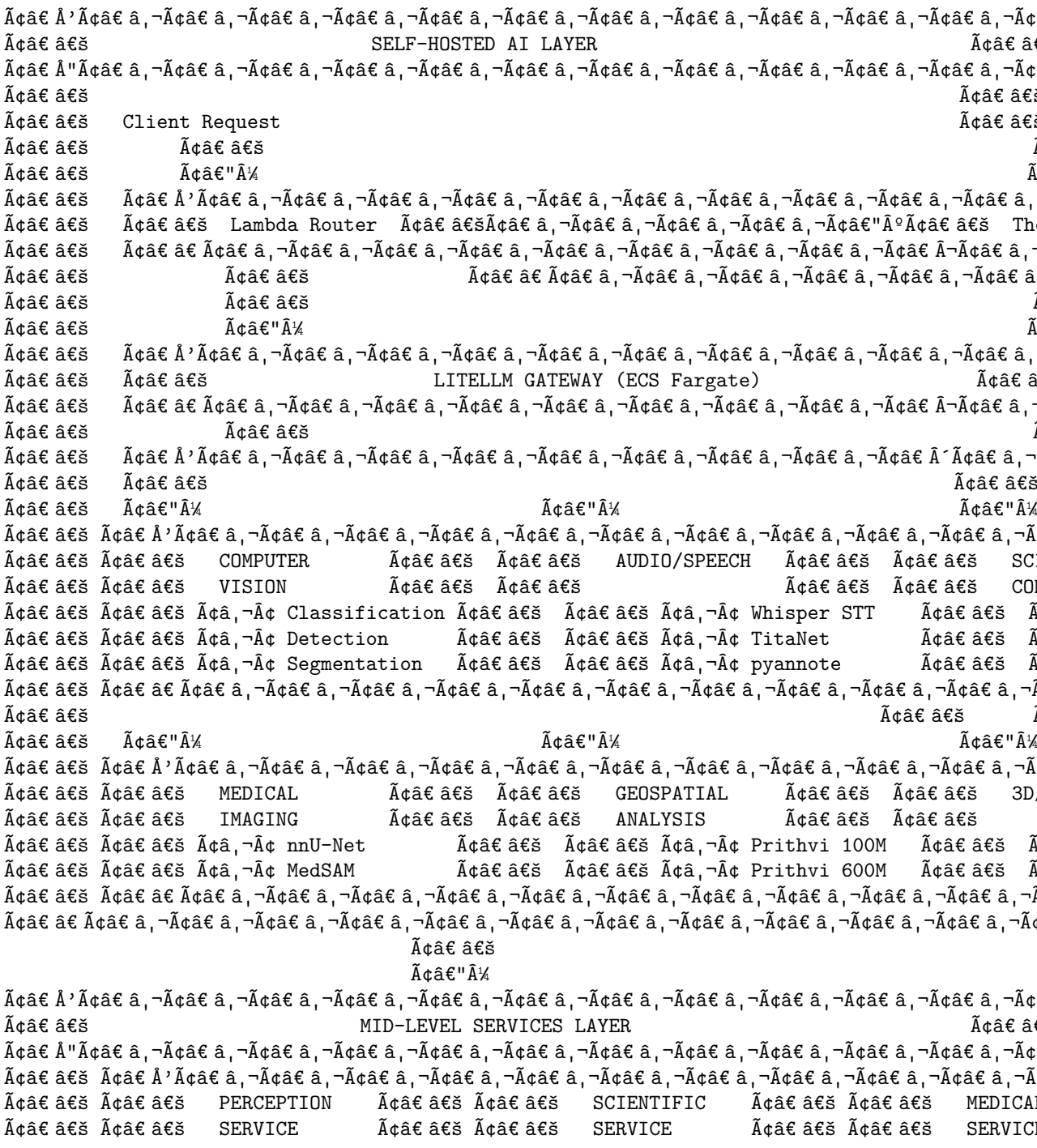
OVERVIEW

This prompt creates the self-hosted AI model configurations and mid-level orchestration services:

1. **Self-Hosted Models** - 30+ AI models deployed on SageMaker across 9 categories
2. **Thermal State Management** - Lambda functions for OFF/COLD/WARM/HOT/AUTOMATIC states
3. **Mid-Level Services** - 5 domain-specific orchestration services
4. **Service State Management** - RUNNING/DEGRADED/DISABLED/OFFLINE states
5. **LiteLLM Configuration** - Self-hosted model routing configuration
6. **Client Notifications** - Real-time model warm-up notifications via WebSocket

All configurations are tier-aware, with self-hosted models requiring Tier 3+ (GROWTH and above).

ARCHITECTURE



```
Ã¢â€ â€š Ã¢â€ â€š Orchestrates:   Ã¢â€ â€š Ã¢â€ â€š Orchestrates:   Ã¢â€ â€š Ã¢â€ â€š Orchestra
Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ YOLOv8/11     Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ AlphaFold2    Ã¢â€ â€š Ã¢â€ â
Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ SAM           Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ ESM-2         Ã¢â€ â€š Ã¢â€ â
Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ CLIP          Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ AlphaGeometry Ã¢â€ â€š Ã¢â€ â
Ã¢â€ â€š Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã
Ã¢â€ â€š Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã
Ã¢â€ â€š Ã¢â€ â€š   GEOSPATIAL    Ã¢â€ â€š Ã¢â€ â€š   3D RECON      Ã¢â€ â€š
Ã¢â€ â€š Ã¢â€ â€š   SERVICE       Ã¢â€ â€š Ã¢â€ â€š   SERVICE       Ã¢â€ â€š
Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ Prithvi       Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ Nerfstudio    Ã¢â€ â€š
Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ SAM           Ã¢â€ â€š Ã¢â€ â€š Ã¢â‚¬Â¢ 3D Gaussian   Ã¢â€ â€š
Ã¢â€ â€š Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
```

DIRECTORY STRUCTURE

```
packages/infrastructure/
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ lib/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ config/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ models/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts                # All model
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ vision.models.ts        # Computer
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ audio.models.ts         # Audio/spe
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ scientific.models.ts    # Scientifi
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ medical.models.ts       # Medical i
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ geospatial.models.ts    # Geospatia
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ generative.models.ts    # 3D/genera
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ services/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts                # All service expo
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ perception.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ scientific.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ medical.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ geospatial.service.ts
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ reconstruction.service.ts
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ lambda/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ thermal/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ manager.ts              # Thermal state ma
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ warmer.ts               # Model warming La
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ notifier.ts             # Client notifica
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ services/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ perception.ts           # Perception orchestratio
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ scientific.ts           # Scientific orchestratio
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ medical.ts              # Medical orchestration
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ geospatial.ts           # Geospatial orchestratio
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ reconstruction.ts       # 3D reconstruction
```

```
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ litellm/
Ã¢â€ â€šÃ¢â€ â€šÃ¢â€ â¸¬Ã¢â€ â¸¬ config/
Ã¢â€ â€šÃ¢â€ â€šÃ¢â€ â¸¬Ã¢â€ â¸¬ self-hosted.yaml              # Self-hosted model conf
Ã¢â€ â€šÃ¢â€ â¸¬Ã¢â€ â¸¬ migrations/
    Ã¢â€ â€šÃ¢â€ â¸¬Ã¢â€ â¸¬ 006_self_hosted_models.sql        # Schema additions
```

PART 1: SELF-HOSTED MODEL DEFINITIONS

AWS SageMaker Instance Pricing (with 75% markup)

Instance	GPUs	VRAM	Base $/hr	Billed $/hr	Best For
ml.g4dn.xlarge	1x T4	16 GB	$0.74	$1.30	Small models, embeddings
ml.g5.xlarge	1x A10G	24 GB	$1.41	$2.47	Medium models
ml.g5.2xlarge	1x A10G	24 GB	$1.52	$2.66	Medium+ models
ml.g5.4xlarge	1x A10G	24 GB	$2.03	$3.55	Large vision models
ml.g5.12xlarge	4x A10G	96 GB	$8.16	$14.28	Large LLMs
ml.g5.48xlarge	8x A10G	192 GB	$20.36	$35.63	Very large models
ml.p4d.24xlarge	8x A100	320 GB	$32.77	$57.35	Scientific computing

packages/infrastructure/lib/config/models/index.ts

```typescript
/**
 * Self-Hosted Model Registry
 * All models available for deployment on SageMaker
 * Pricing includes 75% markup over AWS infrastructure costs
 */

export * from './vision.models';
export * from './audio.models';
export * from './scientific.models';
export * from './medical.models';
export * from './geospatial.models';
export * from './generative.models';
```

```typescript
// ============================================================================
// SHARED TYPES
// ============================================================================

export type ThermalState = 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';
export type ServiceState = 'RUNNING' | 'DEGRADED' | 'DISABLED' | 'OFFLINE';

export interface ThermalConfig {
  defaultState: ThermalState;
  scaleToZeroAfterMinutes: number;
  warmupTimeSeconds: number;
  minInstances: number;
  maxInstances: number;
}

export interface SageMakerModelConfig {
  id: string;
  name: string;
  displayName: string;
  description: string;
  category: ModelCategory;
  specialty: ModelSpecialty;

  // SageMaker Configuration
  image: string;
  instanceType: string;
  environment: Record<string, string>;
  modelDataUrl?: string;

  // Model Capabilities
  parameters: number;
  accuracy?: string;
  benchmark?: string;
  capabilities: string[];
  inputFormats: string[];
  outputFormats: string[];

  // Thermal Management
  thermal: ThermalConfig;

  // Licensing
  license: string;
  licenseUrl?: string;
  commercialUseNotes?: string;

  // Pricing (75% markup over AWS)
```

```
  pricing: SelfHostedModelPricing;

  // Requirements
  minTier: number;
  requiresGPU: boolean;
  gpuMemoryGB: number;

  // Status
  status: 'active' | 'beta' | 'deprecated' | 'coming_soon';
}

export type ModelCategory =
  | 'vision_classification'
  | 'vision_detection'
  | 'vision_segmentation'
  | 'audio_stt'
  | 'audio_speaker'
  | 'scientific_protein'
  | 'scientific_math'
  | 'medical_imaging'
  | 'geospatial'
  | 'generative_3d'
  | 'llm_text';

export type ModelSpecialty =
  | 'image_classification'
  | 'object_detection'
  | 'instance_segmentation'
  | 'speech_to_text'
  | 'speaker_identification'
  | 'protein_folding'
  | 'protein_embeddings'
  | 'geometry_reasoning'
  | 'medical_segmentation'
  | 'satellite_analysis'
  | '3d_reconstruction'
  | 'text_generation';

export interface SelfHostedModelPricing {
  hourlyRate: number;        // Per-hour instance cost (with 75% markup)
  perImage?: number;
  perMinuteAudio?: number;
  perMinuteVideo?: number;
  per3DModel?: number;
  perRequest?: number;
  markup: number;            // 0.75 = 75%
```

```
}

export const INSTANCE_PRICING: Record<string, { base: number; billed: number }> = {
  'ml.g4dn.xlarge':   { base: 0.74,  billed: 1.30 },
  'ml.g5.xlarge':     { base: 1.41,  billed: 2.47 },
  'ml.g5.2xlarge':    { base: 1.52,  billed: 2.66 },
  'ml.g5.4xlarge':    { base: 2.03,  billed: 3.55 },
  'ml.g5.12xlarge':   { base: 8.16,  billed: 14.28 },
  'ml.g5.48xlarge':   { base: 20.36, billed: 35.63 },
  'ml.p4d.24xlarge':  { base: 32.77, billed: 57.35 },
};
```

packages/infrastructure/lib/config/models/vision.models.ts

```
/**
 * Computer Vision Models - Classification, Detection, Segmentation
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

// ============================================================================
// IMAGE CLASSIFICATION MODELS (8 models)
// ============================================================================

export const CLASSIFICATION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'efficientnet-b0',
    name: 'efficientnet-b0',
    displayName: 'EfficientNet-B0',
    description: 'Lightweight image classification model',
    category: 'vision_classification',
    specialty: 'image_classification',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { HF_MODEL_ID: 'google/efficientnet-b0', HF_TASK: 'image-classification' },
    parameters: 5_300_000,
    accuracy: '77.1% ImageNet Top-1',
    capabilities: ['image_classification', 'feature_extraction'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, mi
    license: 'Apache-2.0',
    pricing: { hourlyRate: 1.30, perImage: 0.001, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 2, status: 'active',
  },
  {
```

```
  id: 'efficientnetv2-l',
  name: 'efficientnetv2-l',
  displayName: 'EfficientNetV2-L',
  description: 'State-of-the-art classification with improved training efficiency',
  category: 'vision_classification',
  specialty: 'image_classification',
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.2xlarge',
  environment: { HF_MODEL_ID: 'google/efficientnetv2-l', HF_TASK: 'image-classification' }
  parameters: 118_000_000,
  accuracy: '85.7% ImageNet Top-1',
  capabilities: ['image_classification', 'feature_extraction'],
  inputFormats: ['image/jpeg', 'image/png'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, min
  license: 'Apache-2.0',
  pricing: { hourlyRate: 2.66, perImage: 0.003, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
},
{
  id: 'swin-large',
  name: 'swin-large',
  displayName: 'Swin Transformer Large',
  description: 'Vision Transformer with shifted window attention',
  category: 'vision_classification',
  specialty: 'image_classification',
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.2xlarge',
  environment: { HF_MODEL_ID: 'microsoft/swin-large-patch4-window7-224', HF_TASK: 'image-c
  parameters: 197_000_000,
  accuracy: '87.3% ImageNet Top-1',
  capabilities: ['image_classification', 'feature_extraction'],
  inputFormats: ['image/jpeg', 'image/png'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, min
  license: 'MIT',
  pricing: { hourlyRate: 2.66, perImage: 0.004, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
},
{
  id: 'clip-vit-l14',
  name: 'clip-vit-l14',
  displayName: 'CLIP ViT-L/14',
  description: 'Multimodal vision-language model for zero-shot classification',
  category: 'vision_classification',
  specialty: 'image_classification',
```

```
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { HF_MODEL_ID: 'openai/clip-vit-large-patch14', HF_TASK: 'zero-shot-image-c
    parameters: 428_000_000,
    accuracy: '76.2% ImageNet Zero-Shot',
    capabilities: ['zero_shot_classification', 'image_text_matching'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, min
    license: 'MIT',
    pricing: { hourlyRate: 2.66, perImage: 0.005, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
];

// ============================================================================
// OBJECT DETECTION MODELS (7 models)
// ============================================================================

export const DETECTION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'yolov8n',
    name: 'yolov8n',
    displayName: 'YOLOv8 Nano',
    description: 'Ultra-lightweight real-time object detection',
    category: 'vision_detection',
    specialty: 'object_detection',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { MODEL_NAME: 'yolov8n' },
    parameters: 3_200_000,
    benchmark: '37.3 COCO mAP',
    capabilities: ['object_detection', 'real_time'],
    inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 30, min
    license: 'AGPL-3.0',
    commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
    pricing: { hourlyRate: 1.30, perImage: 0.002, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 2, status: 'active',
  },
  {
    id: 'yolov8m',
    name: 'yolov8m',
    displayName: 'YOLOv8 Medium',
    description: 'Medium model for balanced performance',
```

```
    category: 'vision_detection',
    specialty: 'object_detection',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { MODEL_NAME: 'yolov8m' },
    parameters: 25_900_000,
    benchmark: '50.2 COCO mAP',
    capabilities: ['object_detection', 'real_time'],
    inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, mir
    license: 'AGPL-3.0',
    commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
    pricing: { hourlyRate: 2.47, perImage: 0.004, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 8, status: 'active',
  },
  {
    id: 'yolov8x',
    name: 'yolov8x',
    displayName: 'YOLOv8 Extra Large',
    description: 'Largest YOLOv8 for maximum accuracy',
    category: 'vision_detection',
    specialty: 'object_detection',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { MODEL_NAME: 'yolov8x' },
    parameters: 68_200_000,
    benchmark: '53.9 COCO mAP',
    capabilities: ['object_detection', 'high_accuracy'],
    inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, mir
    license: 'AGPL-3.0',
    commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
    pricing: { hourlyRate: 2.47, perImage: 0.006, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 12, status: 'active',
  },
  {
    id: 'yolov11x',
    name: 'yolov11x',
    displayName: 'YOLOv11 Extra Large',
    description: 'Latest YOLO with improved architecture',
    category: 'vision_detection',
    specialty: 'object_detection',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
```

```
  environment: { MODEL_NAME: 'yolo11x' },
  parameters: 40_000_000,
  benchmark: '54.7 COCO mAP',
  capabilities: ['object_detection', 'real_time', 'high_accuracy'],
  inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, mir
  license: 'AGPL-3.0',
  commercialUseNotes: 'Requires Ultralytics Enterprise License for commercial use',
  pricing: { hourlyRate: 2.47, perImage: 0.006, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
},
{
  id: 'rt-detr-x',
  name: 'rt-detr-x',
  displayName: 'RT-DETR-X',
  description: 'Real-time Detection Transformer (no NMS)',
  category: 'vision_detection',
  specialty: 'object_detection',
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.xlarge',
  environment: { HF_MODEL_ID: 'PekingU/rtdetr_r101vd', HF_TASK: 'object-detection' },
  parameters: 40_000_000,
  benchmark: '54.8 COCO mAP',
  capabilities: ['object_detection', 'real_time', 'end_to_end'],
  inputFormats: ['image/jpeg', 'image/png'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, mir
  license: 'Apache-2.0',
  pricing: { hourlyRate: 2.47, perImage: 0.005, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
},
{
  id: 'grounding-dino',
  name: 'grounding-dino',
  displayName: 'Grounding DINO',
  description: 'Open-vocabulary object detection with text prompts',
  category: 'vision_detection',
  specialty: 'object_detection',
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.2xlarge',
  environment: { HF_MODEL_ID: 'IDEA-Research/grounding-dino-base', HF_TASK: 'zero-shot-obj
  parameters: 172_000_000,
  benchmark: '52.5 COCO Zero-Shot',
  capabilities: ['zero_shot_detection', 'text_prompted', 'open_vocabulary'],
  inputFormats: ['image/jpeg', 'image/png'],
```

```
    outputFormats: ['application/json'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, mir
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.66, perImage: 0.008, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
];

// ============================================================================
// SEGMENTATION MODELS (4 models)
// ============================================================================

export const SEGMENTATION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'sam-vit-h',
    name: 'sam-vit-h',
    displayName: 'SAM ViT-H',
    description: 'Segment Anything Model - largest variant',
    category: 'vision_segmentation',
    specialty: 'instance_segmentation',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { HF_MODEL_ID: 'facebook/sam-vit-huge', HF_TASK: 'mask-generation' },
    parameters: 636_000_000,
    capabilities: ['instance_segmentation', 'interactive', 'zero_shot'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json', 'image/png'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 120, mi
    license: 'Apache-2.0',
    pricing: { hourlyRate: 3.55, perImage: 0.015, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 20, status: 'active',
  },
  {
    id: 'sam2',
    name: 'sam2',
    displayName: 'SAM 2',
    description: 'Segment Anything Model 2 with video support',
    category: 'vision_segmentation',
    specialty: 'instance_segmentation',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { MODEL_NAME: 'sam2_hiera_large' },
    parameters: 224_000_000,
    capabilities: ['instance_segmentation', 'video_segmentation', 'interactive', 'zero_shot
    inputFormats: ['image/jpeg', 'image/png', 'video/mp4'],
    outputFormats: ['application/json', 'image/png', 'video/mp4'],
```

```
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 120, mi
    license: 'Apache-2.0',
    pricing: { hourlyRate: 3.55, perImage: 0.012, perMinuteVideo: 0.50, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
  {
    id: 'mobilesam',
    name: 'mobilesam',
    displayName: 'MobileSAM',
    description: 'Lightweight SAM for fast inference',
    category: 'vision_segmentation',
    specialty: 'instance_segmentation',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { HF_MODEL_ID: 'dhkim2810/mobilesam', HF_TASK: 'mask-generation' },
    parameters: 10_000_000,
    capabilities: ['instance_segmentation', 'interactive', 'fast'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json', 'image/png'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 30, min
    license: 'Apache-2.0',
    pricing: { hourlyRate: 1.30, perImage: 0.004, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 4, status: 'active',
  },
  {
    id: 'mask-rcnn',
    name: 'mask-rcnn',
    displayName: 'Mask R-CNN',
    description: 'Classic instance segmentation model',
    category: 'vision_segmentation',
    specialty: 'instance_segmentation',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { MODEL_NAME: 'mask_rcnn_R_101_FPN_3x', DETECTRON2_CONFIG: 'COCO-InstanceSe
    parameters: 63_000_000,
    benchmark: '42.9 COCO AP',
    capabilities: ['instance_segmentation', 'object_detection'],
    inputFormats: ['image/jpeg', 'image/png'],
    outputFormats: ['application/json', 'image/png'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, min
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.47, perImage: 0.006, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 8, status: 'active',
  },
];
```

```
// Export all vision models
export const ALL_VISION_MODELS = [...CLASSIFICATION_MODELS, ...DETECTION_MODELS, ...SEGMENTA
```

packages/infrastructure/lib/config/models/audio.models.ts

```
/**
 * Audio & Speech Models - STT, Speaker Recognition
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

// ============================================================================
// SPEECH-TO-TEXT MODELS (3 models)
// ============================================================================

export const STT_MODELS: SageMakerModelConfig[] = [
  {
    id: 'parakeet-tdt-1b',
    name: 'parakeet-tdt-1b',
    displayName: 'NVIDIA Parakeet TDT 1.1B',
    description: 'State-of-the-art ASR with 4.4% WER on LibriSpeech',
    category: 'audio_stt',
    specialty: 'speech_to_text',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { MODEL_NAME: 'nvidia/parakeet-tdt-1.1b' },
    parameters: 1_100_000_000,
    accuracy: '4.4% WER LibriSpeech test-clean',
    capabilities: ['speech_to_text', 'english', 'real_time'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, mir
    license: 'CC-BY-4.0',
    pricing: { hourlyRate: 2.47, perMinuteAudio: 0.03, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
    id: 'whisper-large-v3',
    name: 'whisper-large-v3',
    displayName: 'OpenAI Whisper Large V3',
    description: 'Multilingual speech recognition and translation',
    category: 'audio_stt',
    specialty: 'speech_to_text',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { HF_MODEL_ID: 'openai/whisper-large-v3', HF_TASK: 'automatic-speech-recogn
```

```
    parameters: 1_550_000_000,
    accuracy: '5.0% WER LibriSpeech test-clean',
    capabilities: ['speech_to_text', 'multilingual', 'translation', 'timestamps'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac', 'audio/m4a'],
    outputFormats: ['application/json', 'text/plain', 'text/vtt', 'text/srt'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, mir
    license: 'MIT',
    pricing: { hourlyRate: 2.47, perMinuteAudio: 0.04, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
    id: 'whisper-turbo',
    name: 'whisper-turbo',
    displayName: 'OpenAI Whisper Turbo',
    description: 'Fast multilingual ASR with 8x speed improvement',
    category: 'audio_stt',
    specialty: 'speech_to_text',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
    environment: { HF_MODEL_ID: 'openai/whisper-large-v3-turbo', HF_TASK: 'automatic-speech-
    parameters: 809_000_000,
    accuracy: '5.5% WER LibriSpeech test-clean',
    capabilities: ['speech_to_text', 'multilingual', 'fast', 'real_time'],
    inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac', 'audio/m4a'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, mir
    license: 'MIT',
    pricing: { hourlyRate: 1.30, perMinuteAudio: 0.02, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 6, status: 'active',
  },
];

// ============================================================================
// SPEAKER RECOGNITION MODELS (3 models)
// ============================================================================

export const SPEAKER_MODELS: SageMakerModelConfig[] = [
  {
    id: 'titanet-large',
    name: 'titanet-large',
    displayName: 'NVIDIA TitaNet-Large',
    description: 'Speaker verification and embedding extraction',
    category: 'audio_speaker',
    specialty: 'speaker_identification',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g4dn.xlarge',
```

```
  environment: { MODEL_NAME: 'nvidia/speakerverification_en_titanet_large' },
  parameters: 23_000_000,
  accuracy: '0.68% EER VoxCeleb1',
  capabilities: ['speaker_verification', 'speaker_embedding', 'speaker_identification'],
  inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, mir
  license: 'CC-BY-4.0',
  pricing: { hourlyRate: 1.30, perMinuteAudio: 0.02, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 4, status: 'active',
},
{
  id: 'ecapa-tdnn',
  name: 'ecapa-tdnn',
  displayName: 'ECAPA-TDNN',
  description: 'Emphasized channel attention for speaker verification',
  category: 'audio_speaker',
  specialty: 'speaker_identification',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g4dn.xlarge',
  environment: { MODEL_NAME: 'speechbrain/spkrec-ecapa-voxceleb' },
  parameters: 20_800_000,
  accuracy: '0.80% EER VoxCeleb1',
  capabilities: ['speaker_verification', 'speaker_embedding'],
  inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 45, mir
  license: 'Apache-2.0',
  pricing: { hourlyRate: 1.30, perMinuteAudio: 0.02, markup: 0.75 },
  minTier: 3, requiresGPU: true, gpuMemoryGB: 4, status: 'active',
},
{
  id: 'pyannote-speaker-diarization',
  name: 'pyannote-speaker-diarization',
  displayName: 'pyannote Speaker Diarization 3.1',
  description: 'Who spoke when - speaker diarization pipeline',
  category: 'audio_speaker',
  specialty: 'speaker_identification',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.xlarge',
  environment: { MODEL_NAME: 'pyannote/speaker-diarization-3.1' },
  parameters: 0,
  accuracy: '18.4% DER AMI Mix-Headset',
  capabilities: ['speaker_diarization', 'speaker_counting', 'overlap_detection'],
  inputFormats: ['audio/wav', 'audio/mp3', 'audio/flac'],
  outputFormats: ['application/json', 'text/rttm'],
```

```
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 60, mir
    license: 'MIT',
    commercialUseNotes: 'Requires pyannote/speaker-diarization access grant',
    pricing: { hourlyRate: 2.47, perMinuteAudio: 0.05, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 8, status: 'active',
  },
];

export const ALL_AUDIO_MODELS = [...STT_MODELS, ...SPEAKER_MODELS];
```

packages/infrastructure/lib/config/models/scientific.models.ts

```
/**
 * Scientific Computing Models - Protein folding, embeddings, math reasoning
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const SCIENTIFIC_MODELS: SageMakerModelConfig[] = [
  {
    id: 'alphafold2',
    name: 'alphafold2',
    displayName: 'AlphaFold 2',
    description: 'Protein structure prediction with near-experimental accuracy',
    category: 'scientific_protein',
    specialty: 'protein_folding',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.12xlarge',
    environment: { MODEL_NAME: 'alphafold2_ptm', JAX_PLATFORM_NAME: 'gpu' },
    parameters: 93_000_000,
    accuracy: 'GDT > 90 on CASP14',
    capabilities: ['protein_folding', 'structure_prediction', 'confidence_estimation'],
    inputFormats: ['text/fasta', 'text/plain'],
    outputFormats: ['application/pdb', 'application/mmcif', 'application/json'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 180, mir
    license: 'Apache-2.0',
    pricing: { hourlyRate: 14.28, perRequest: 2.00, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 96, status: 'active',
  },
  {
    id: 'esm2-3b',
    name: 'esm2-3b',
    displayName: 'ESM-2 3B',
    description: 'Protein language model for embeddings and analysis',
    category: 'scientific_protein',
    specialty: 'protein_embeddings',
```

```
  image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.4xlarge',
  environment: { HF_MODEL_ID: 'facebook/esm2_t36_3B_UR50D' },
  parameters: 3_000_000_000,
  capabilities: ['protein_embedding', 'sequence_analysis', 'structure_prediction'],
  inputFormats: ['text/fasta', 'text/plain'],
  outputFormats: ['application/json'],
  thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, mir
  license: 'MIT',
  pricing: { hourlyRate: 3.55, perRequest: 0.50, markup: 0.75 },
  minTier: 4, requiresGPU: true, gpuMemoryGB: 20, status: 'active',
},
{
  id: 'alphageometry',
  name: 'alphageometry',
  displayName: 'AlphaGeometry',
  description: 'Olympiad-level geometry problem solver',
  category: 'scientific_math',
  specialty: 'geometry_reasoning',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.2xlarge',
  environment: { MODEL_NAME: 'alphageometry' },
  parameters: 150_000_000,
  accuracy: '25/30 IMO geometry problems',
  capabilities: ['geometry_solving', 'proof_generation', 'theorem_proving'],
  inputFormats: ['application/json', 'text/plain'],
  outputFormats: ['application/json', 'text/plain'],
  thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 90, minI
  license: 'Apache-2.0',
  pricing: { hourlyRate: 2.66, perRequest: 0.30, markup: 0.75 },
  minTier: 4, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
},
{
  id: 'protenix',
  name: 'protenix',
  displayName: 'Protenix',
  description: 'Open-source AlphaFold3-like structure prediction',
  category: 'scientific_protein',
  specialty: 'protein_folding',
  image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
  instanceType: 'ml.g5.12xlarge',
  environment: { MODEL_NAME: 'protenix' },
  parameters: 0,
  capabilities: ['protein_folding', 'dna_rna_binding', 'ligand_binding', 'multi_chain'],
  inputFormats: ['text/fasta', 'application/json'],
  outputFormats: ['application/pdb', 'application/mmcif', 'application/json'],
```

```
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 180, mir
    license: 'Apache-2.0',
    pricing: { hourlyRate: 14.28, perRequest: 2.50, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 96, status: 'beta',
  },
];
```

packages/infrastructure/lib/config/models/medical.models.ts

```
/**
 * Medical Imaging Models - HIPAA-compliant medical image analysis
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const MEDICAL_MODELS: SageMakerModelConfig[] = [
  {
    id: 'nnunet',
    name: 'nnunet',
    displayName: 'nnU-Net',
    description: 'Self-configuring medical image segmentation',
    category: 'medical_imaging',
    specialty: 'medical_segmentation',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
    environment: { MODEL_NAME: 'nnunet', NNUNET_DATASET: 'generic' },
    parameters: 31_000_000,
    accuracy: 'State-of-the-art on 23/23 Medical Segmentation Decathlon tasks',
    capabilities: ['medical_segmentation', 'tumor_detection', 'organ_segmentation', '3d_imag
    inputFormats: ['application/dicom', 'application/nifti', 'image/png'],
    outputFormats: ['application/nifti', 'application/json', 'image/png'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, mir
    license: 'Apache-2.0',
    commercialUseNotes: 'HIPAA compliant when deployed in compliant AWS environment',
    pricing: { hourlyRate: 2.66, perImage: 0.10, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
  {
    id: 'medsam',
    name: 'medsam',
    displayName: 'MedSAM',
    description: 'Segment Anything Model fine-tuned for medical images',
    category: 'medical_imaging',
    specialty: 'medical_segmentation',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.2xlarge',
```

```
    environment: { HF_MODEL_ID: 'wanglab/medsam-vit-base', HF_TASK: 'mask-generation' },
    parameters: 93_000_000,
    capabilities: ['medical_segmentation', 'interactive', 'multi_modality'],
    inputFormats: ['application/dicom', 'application/nifti', 'image/png', 'image/jpeg'],
    outputFormats: ['application/json', 'image/png'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 90, minI
    license: 'Apache-2.0',
    commercialUseNotes: 'HIPAA compliant when deployed in compliant AWS environment',
    pricing: { hourlyRate: 2.66, perImage: 0.08, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 12, status: 'active',
  },
];
```

packages/infrastructure/lib/config/models/geospatial.models.ts

```
/**
 * Geospatial Analysis Models - Satellite imagery and earth observation
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const GEOSPATIAL_MODELS: SageMakerModelConfig[] = [
  {
    id: 'prithvi-100m',
    name: 'prithvi-100m',
    displayName: 'Prithvi 100M',
    description: 'NASA/IBM geospatial foundation model (100M parameters)',
    category: 'geospatial',
    specialty: 'satellite_analysis',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { HF_MODEL_ID: 'ibm-nasa-geospatial/Prithvi-100M' },
    parameters: 100_000_000,
    capabilities: ['satellite_analysis', 'land_use', 'flood_detection', 'crop_mapping'],
    inputFormats: ['image/tiff', 'image/geotiff', 'image/png'],
    outputFormats: ['application/json', 'image/geotiff'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 90, minI
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.47, perImage: 0.05, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 10, status: 'active',
  },
  {
    id: 'prithvi-600m',
    name: 'prithvi-600m',
    displayName: 'Prithvi 600M',
    description: 'NASA/IBM geospatial foundation model (600M parameters)',
```

```
    category: 'geospatial',
    specialty: 'satellite_analysis',
    image: 'pytorch-inference:2.1-transformers4.36-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { HF_MODEL_ID: 'ibm-nasa-geospatial/Prithvi-EO-2.0-600M' },
    parameters: 600_000_000,
    capabilities: ['satellite_analysis', 'land_use', 'flood_detection', 'crop_mapping', 'cha
    inputFormats: ['image/tiff', 'image/geotiff', 'image/png'],
    outputFormats: ['application/json', 'image/geotiff'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, mir
    license: 'Apache-2.0',
    pricing: { hourlyRate: 3.55, perImage: 0.10, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 20, status: 'active',
  },
];
```

packages/infrastructure/lib/config/models/generative.models.ts

```
/**
 * 3D & Generative Models - 3D reconstruction, self-hosted LLMs
 */

import { SageMakerModelConfig, INSTANCE_PRICING } from './index';

export const GENERATIVE_3D_MODELS: SageMakerModelConfig[] = [
  {
    id: 'nerfstudio',
    name: 'nerfstudio',
    displayName: 'Nerfstudio',
    description: 'Neural Radiance Fields for 3D scene reconstruction',
    category: 'generative_3d',
    specialty: '3d_reconstruction',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { MODEL_NAME: 'nerfstudio', NERFSTUDIO_METHOD: 'nerfacto' },
    parameters: 0,
    capabilities: ['3d_reconstruction', 'novel_view_synthesis', 'scene_capture'],
    inputFormats: ['video/mp4', 'image/jpeg', 'image/png'],
    outputFormats: ['model/gltf+json', 'model/obj', 'video/mp4'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, mir
    license: 'Apache-2.0',
    pricing: { hourlyRate: 3.55, per3DModel: 5.00, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 24, status: 'active',
  },
  {
    id: '3d-gaussian-splatting',
```

```
    name: '3d-gaussian-splatting',
    displayName: '3D Gaussian Splatting',
    description: 'Real-time radiance field rendering with 3D Gaussians',
    category: 'generative_3d',
    specialty: '3d_reconstruction',
    image: 'pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.4xlarge',
    environment: { MODEL_NAME: '3dgs' },
    parameters: 0,
    capabilities: ['3d_reconstruction', 'real_time_rendering', 'novel_view_synthesis'],
    inputFormats: ['video/mp4', 'image/jpeg', 'image/png'],
    outputFormats: ['model/ply', 'model/gltf+json', 'video/mp4'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 120, mir
    license: 'Custom',
    commercialUseNotes: 'Check license for commercial use terms',
    pricing: { hourlyRate: 3.55, per3DModel: 4.00, markup: 0.75 },
    minTier: 4, requiresGPU: true, gpuMemoryGB: 24, status: 'active',
  },
];

export const TEXT_GENERATION_MODELS: SageMakerModelConfig[] = [
  {
    id: 'mistral-7b-instruct',
    name: 'mistral-7b-instruct',
    displayName: 'Mistral 7B Instruct',
    description: 'Efficient instruction-following model',
    category: 'llm_text',
    specialty: 'text_generation',
    image: 'huggingface-pytorch-tgi-inference:2.1-tgi1.4-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.xlarge',
    environment: { HF_MODEL_ID: 'mistralai/Mistral-7B-Instruct-v0.3', MAX_INPUT_LENGTH: '409
    parameters: 7_000_000_000,
    capabilities: ['text_generation', 'instruction_following', 'function_calling'],
    inputFormats: ['application/json', 'text/plain'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'COLD', scaleToZeroAfterMinutes: 15, warmupTimeSeconds: 90, mir
    license: 'Apache-2.0',
    pricing: { hourlyRate: 2.47, perRequest: 0.005, markup: 0.75 },
    minTier: 3, requiresGPU: true, gpuMemoryGB: 16, status: 'active',
  },
  {
    id: 'llama-3-70b-instruct',
    name: 'llama-3-70b-instruct',
    displayName: 'Llama 3 70B Instruct',
    description: 'Large open-weight model for complex tasks',
    category: 'llm_text',
```

```
    specialty: 'text_generation',
    image: 'huggingface-pytorch-tgi-inference:2.1-tgi1.4-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.48xlarge',
    environment: { HF_MODEL_ID: 'meta-llama/Meta-Llama-3-70B-Instruct', MAX_INPUT_LENGTH: '8
    parameters: 70_000_000_000,
    capabilities: ['text_generation', 'instruction_following', 'reasoning', 'code_generation
    inputFormats: ['application/json', 'text/plain'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 300, min
    license: 'Llama 3 Community License',
    commercialUseNotes: 'Free for commercial use, attribution required',
    pricing: { hourlyRate: 35.63, perRequest: 0.05, markup: 0.75 },
    minTier: 5, requiresGPU: true, gpuMemoryGB: 160, status: 'active',
  },
  {
    id: 'qwen-72b-instruct',
    name: 'qwen-72b-instruct',
    displayName: 'Qwen 2.5 72B Instruct',
    description: 'State-of-the-art multilingual model',
    category: 'llm_text',
    specialty: 'text_generation',
    image: 'huggingface-pytorch-tgi-inference:2.1-tgi1.4-gpu-py310-cu121-ubuntu22.04',
    instanceType: 'ml.g5.48xlarge',
    environment: { HF_MODEL_ID: 'Qwen/Qwen2.5-72B-Instruct', MAX_INPUT_LENGTH: '32768', MAX_
    parameters: 72_000_000_000,
    capabilities: ['text_generation', 'instruction_following', 'multilingual', 'long_context
    inputFormats: ['application/json', 'text/plain'],
    outputFormats: ['application/json', 'text/plain'],
    thermal: { defaultState: 'OFF', scaleToZeroAfterMinutes: 10, warmupTimeSeconds: 300, min
    license: 'Apache-2.0',
    pricing: { hourlyRate: 35.63, perRequest: 0.05, markup: 0.75 },
    minTier: 5, requiresGPU: true, gpuMemoryGB: 160, status: 'active',
  },
];

export const ALL_GENERATIVE_MODELS = [...GENERATIVE_3D_MODELS, ...TEXT_GENERATION_MODELS];
```

PART 2: MID-LEVEL SERVICES

packages/infrastructure/lib/config/services/index.ts

```
/**
 * Mid-Level Service Definitions
 * Services that orchestrate multiple AI models for domain-specific tasks
 */
```

```typescript
export type ServiceState = 'RUNNING' | 'DEGRADED' | 'DISABLED' | 'OFFLINE';

export interface MidLevelServiceConfig {
  id: string;
  name: string;
  displayName: string;
  description: string;
  requiredModels: string[];
  optionalModels: string[];
  defaultState: ServiceState;
  gracefulDegradation: boolean;
  pricing: ServicePricing;
  minTier: number;
  endpoints: ServiceEndpoint[];
}

export interface ServicePricing {
  perRequest?: number;
  perMinuteAudio?: number;
  perMinuteVideo?: number;
  perImage?: number;
  per3DModel?: number;
  markup: number;
}

export interface ServiceEndpoint {
  path: string;
  method: 'GET' | 'POST' | 'PUT' | 'DELETE';
  description: string;
  requiredModels: string[];
  inputFormats: string[];
  outputFormats: string[];
}

export const SERVICE_STATE_COLORS: Record<ServiceState, string> = {
  RUNNING: '#22c55e',    // green
  DEGRADED: '#f59e0b',   // yellow
  DISABLED: '#6b7280',   // gray
  OFFLINE: '#ef4444',    // red
};
```

packages/infrastructure/lib/config/services/perception.service.ts

```typescript
/**
 * Perception Service - Unified computer vision pipeline
```

```
 */

import { MidLevelServiceConfig } from './index';

export const PERCEPTION_SERVICE: MidLevelServiceConfig = {
  id: 'perception',
  name: 'perception',
  displayName: 'Perception Service',
  description: 'Unified computer vision pipeline for detection, segmentation, and classifica
  requiredModels: ['yolov8m', 'mobilesam'],
  optionalModels: ['yolov8x', 'yolov11x', 'sam-vit-h', 'sam2', 'clip-vit-l14', 'grounding-di
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { perImage: 0.02, perMinuteVideo: 0.50, markup: 0.40 },
  minTier: 3,
  endpoints: [
    { path: '/perception/detect', method: 'POST', description: 'Detect objects in images', r
    { path: '/perception/segment', method: 'POST', description: 'Segment objects or regions'
    { path: '/perception/classify', method: 'POST', description: 'Classify images', required
    { path: '/perception/analyze', method: 'POST', description: 'Full perception pipeline',
  ],
};
```

packages/infrastructure/lib/config/services/scientific.service.ts

```
/**
 * Scientific Computing Service - Protein analysis and math reasoning
 */

import { MidLevelServiceConfig } from './index';

export const SCIENTIFIC_SERVICE: MidLevelServiceConfig = {
  id: 'scientific',
  name: 'scientific',
  displayName: 'Scientific Computing Service',
  description: 'Protein folding, embeddings, and computational biology pipelines',
  requiredModels: ['esm2-3b'],
  optionalModels: ['alphafold2', 'alphageometry', 'protenix'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { perRequest: 0.50, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/scientific/protein/embed', method: 'POST', description: 'Generate protein embe
    { path: '/scientific/protein/fold', method: 'POST', description: 'Predict protein 3D str
    { path: '/scientific/geometry/solve', method: 'POST', description: 'Solve geometry probl
```

```
  ],
};
```

packages/infrastructure/lib/config/services/medical.service.ts

```
/**
 * Medical Imaging Service - HIPAA-compliant medical analysis
 */

import { MidLevelServiceConfig } from './index';

export const MEDICAL_SERVICE: MidLevelServiceConfig = {
  id: 'medical',
  name: 'medical',
  displayName: 'Medical Imaging Service',
  description: 'HIPAA-compliant medical image segmentation and analysis',
  requiredModels: ['medsam'],
  optionalModels: ['nnunet', 'whisper-large-v3'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { perImage: 0.15, perMinuteAudio: 0.08, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/medical/segment', method: 'POST', description: 'Segment anatomical structures'
    { path: '/medical/segment/3d', method: 'POST', description: 'Volumetric 3D segmentation'
    { path: '/medical/transcribe', method: 'POST', description: 'Transcribe medical dictatio
  ],
};
```

packages/infrastructure/lib/config/services/geospatial.service.ts

```
/**
 * Geospatial Analysis Service - Satellite imagery analysis
 */

import { MidLevelServiceConfig } from './index';

export const GEOSPATIAL_SERVICE: MidLevelServiceConfig = {
  id: 'geospatial',
  name: 'geospatial',
  displayName: 'Geospatial Analysis Service',
  description: 'Satellite imagery analysis for land use, flood detection, and crop mapping',
  requiredModels: ['prithvi-100m'],
  optionalModels: ['prithvi-600m', 'mobilesam'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
```

```
  pricing: { perImage: 0.08, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/geospatial/classify', method: 'POST', description: 'Classify land use', requi
    { path: '/geospatial/detect/floods', method: 'POST', description: 'Detect flood extent',
    { path: '/geospatial/change', method: 'POST', description: 'Detect changes between image
  ],
};
```

packages/infrastructure/lib/config/services/reconstruction.service.ts

```
/**
 * 3D Reconstruction Service - NeRF and 3D Gaussian Splatting
 */

import { MidLevelServiceConfig } from './index';

export const RECONSTRUCTION_SERVICE: MidLevelServiceConfig = {
  id: 'reconstruction',
  name: 'reconstruction',
  displayName: '3D Reconstruction Service',
  description: 'Generate 3D models from images and videos using neural radiance fields',
  requiredModels: ['nerfstudio'],
  optionalModels: ['3d-gaussian-splatting'],
  defaultState: 'DISABLED',
  gracefulDegradation: true,
  pricing: { per3DModel: 8.00, markup: 0.40 },
  minTier: 4,
  endpoints: [
    { path: '/reconstruction/nerf', method: 'POST', description: 'Create 3D model using NeRF
    { path: '/reconstruction/gaussian', method: 'POST', description: 'Fast 3D using Gaussian
    { path: '/reconstruction/render', method: 'POST', description: 'Render novel views', rec
  ],
};
```

PART 3: THERMAL STATE MANAGEMENT

Thermal State Definitions

State	Description	Instance Count	Use Case
OFF	Completely disabled	0	Not needed, cost savings
COLD	Scales to zero when idle	0 (on-demand)	Infrequent use
WARM	Minimum instances always running	min (1+)	Regular use
HOT	Pre-scaled for high traffic	min + buffer	High demand

State	Description	Instance Count	Use Case
AUTOMATIC	AI-managed scaling	dynamic	Variable workloads

packages/infrastructure/lambda/thermal/manager.ts (Summary)

```
/**
 * Thermal State Manager Lambda
 *
 * Key Functions:
 * - handleListModels(): List all models with thermal states
 * - handleGetModel(): Get specific model thermal state + metrics
 * - handleUpdateState(): Admin update thermal state
 * - handleBulkUpdate(): Bulk update multiple models
 * - handleWarmModel(): Request model warm-up (API users)
 * - handleScheduledCheck(): Auto-scaling and scale-to-zero logic
 *
 * Scheduled Tasks (every 5 minutes):
 * - Check transition status
 * - Handle AUTOMATIC scaling based on metrics
 * - Scale to zero inactive WARM/HOT models
 */

// API Routes
// GET  /thermal/models          - List all models
// GET  /thermal/models/:id      - Get model details
// PUT  /thermal/models/:id      - Update thermal state
// POST /thermal/bulk            - Bulk update
// POST /thermal/warm/:id        - Request warm-up
// GET  /thermal/metrics         - Get summary metrics
```

packages/infrastructure/lambda/thermal/notifier.ts (Summary)

```
/**
 * Client Notification Lambda
 *
 * Sends real-time notifications to clients via WebSocket when:
 * - Model warm-up starts
 * - Model becomes ready
 * - Model goes cold/offline
 * - Service state changes
 */
```

PART 4: DATABASE SCHEMA

packages/infrastructure/migrations/006_self_hosted_models.sql

```sql
-- Self-Hosted Model Registry
CREATE TABLE IF NOT EXISTS self_hosted_models (
    id VARCHAR(100) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    display_name VARCHAR(255) NOT NULL,
    description TEXT,
    category VARCHAR(50) NOT NULL,
    specialty VARCHAR(50) NOT NULL,
    sagemaker_image VARCHAR(500) NOT NULL,
    instance_type VARCHAR(50) NOT NULL,
    environment JSONB NOT NULL DEFAULT '{}',
    parameters BIGINT,
    accuracy VARCHAR(100),
    capabilities TEXT[] NOT NULL DEFAULT '{}',
    input_formats TEXT[] NOT NULL DEFAULT '{}',
    output_formats TEXT[] NOT NULL DEFAULT '{}',
    license VARCHAR(100) NOT NULL,
    commercial_use_notes TEXT,
    hourly_rate DECIMAL(10,4) NOT NULL,
    per_image DECIMAL(10,6),
    per_minute_audio DECIMAL(10,6),
    per_3d_model DECIMAL(10,4),
    markup_percent DECIMAL(5,2) NOT NULL DEFAULT 75.00,
    min_tier INTEGER NOT NULL DEFAULT 3,
    requires_gpu BOOLEAN NOT NULL DEFAULT true,
    gpu_memory_gb INTEGER NOT NULL,
    status VARCHAR(20) NOT NULL DEFAULT 'active',
    created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
);

-- Thermal States per app/environment
CREATE TABLE IF NOT EXISTS thermal_states (
    app_id VARCHAR(100) NOT NULL,
    environment VARCHAR(20) NOT NULL,
    model_id VARCHAR(100) NOT NULL REFERENCES self_hosted_models(id),
    current_state VARCHAR(20) NOT NULL DEFAULT 'COLD',
    target_state VARCHAR(20) NOT NULL DEFAULT 'COLD',
    instance_count INTEGER NOT NULL DEFAULT 0,
    min_instances INTEGER NOT NULL DEFAULT 0,
    max_instances INTEGER NOT NULL DEFAULT 3,
    last_activity TIMESTAMP WITH TIME ZONE,
```

```
    last_state_change TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    scale_to_zero_after_minutes INTEGER NOT NULL DEFAULT 15,
    warmup_time_seconds INTEGER NOT NULL DEFAULT 60,
    is_transitioning BOOLEAN NOT NULL DEFAULT false,
    error_message TEXT,
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    updated_by VARCHAR(100),
    PRIMARY KEY (app_id, environment, model_id),
    CONSTRAINT valid_thermal_state CHECK (current_state IN ('OFF', 'COLD', 'WARM', 'HOT', 'A
);

-- Mid-Level Services
CREATE TABLE IF NOT EXISTS mid_level_services (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    display_name VARCHAR(255) NOT NULL,
    description TEXT,
    required_models TEXT[] NOT NULL DEFAULT '{}',
    optional_models TEXT[] NOT NULL DEFAULT '{}',
    default_state VARCHAR(20) NOT NULL DEFAULT 'DISABLED',
    graceful_degradation BOOLEAN NOT NULL DEFAULT true,
    per_request DECIMAL(10,6),
    per_image DECIMAL(10,6),
    per_3d_model DECIMAL(10,4),
    markup_percent DECIMAL(5,2) NOT NULL DEFAULT 40.00,
    min_tier INTEGER NOT NULL DEFAULT 3,
    created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
);

-- Service States per app/environment
CREATE TABLE IF NOT EXISTS service_states (
    app_id VARCHAR(100) NOT NULL,
    environment VARCHAR(20) NOT NULL,
    service_id VARCHAR(50) NOT NULL REFERENCES mid_level_services(id),
    current_state VARCHAR(20) NOT NULL DEFAULT 'DISABLED',
    degraded_reason TEXT,
    available_models TEXT[] NOT NULL DEFAULT '{}',
    unavailable_models TEXT[] NOT NULL DEFAULT '{}',
    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
    PRIMARY KEY (app_id, environment, service_id),
    CONSTRAINT valid_service_state CHECK (current_state IN ('RUNNING', 'DEGRADED', 'DISABLEI
);

-- Model Usage Tracking (for billing)
CREATE TABLE IF NOT EXISTS model_usage (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    app_id VARCHAR(100) NOT NULL,
    environment VARCHAR(20) NOT NULL,
    tenant_id VARCHAR(100) NOT NULL,
    user_id VARCHAR(100),
    model_id VARCHAR(100) NOT NULL,
    service_id VARCHAR(50),
    operation VARCHAR(100) NOT NULL,
    processing_time_ms INTEGER NOT NULL,
    input_size_bytes INTEGER,
    output_size_bytes INTEGER,
    computed_cost DECIMAL(12,8) NOT NULL,
    request_id VARCHAR(100),
    metadata JSONB,
    created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_model_usage_tenant ON model_usage(tenant_id);
CREATE INDEX idx_model_usage_model ON model_usage(model_id);
CREATE INDEX idx_model_usage_created ON model_usage(created_at);
```

PART 5: ESTIMATED COSTS BY TIER

Self-Hosted Model Costs (Monthly)

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
SageMaker (Vision)	N/A	N/A	~$150	~$400	~$1,000
SageMaker (Audio)	N/A	N/A	~$100	~$250	~$600
SageMaker (Scientific)	N/A	N/A	N/A	~$300	~$800
SageMaker (Medical)	N/A	N/A	N/A	~$200	~$500
SageMaker (Geospatial)	N/A	N/A	N/A	~$150	~$400
SageMaker (3D)	N/A	N/A	N/A	~$200	~$500
SageMaker (LLMs)	N/A	N/A	~$100	~$500	~$2,000
Lambda (Thermal/Services)	N/A	N/A	~$30	~$80	~$250

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
DynamoDB (States)	N/A	N/A	~$5	~$15	~$50
Prompt 6 Total	**N/A**	**N/A**	**~$385**	**~$2,095**	**~$6,100**

Notes: - Self-hosted models require Tier 3+ (GROWTH tier or above) - Costs assume models in COLD state (scale to zero when idle) - 75% markup on SageMaker costs included for billing

---

MODEL SUMMARY BY CATEGORY

Computer Vision (19 models)

Category	Count	Models
Classification	8	EfficientNet-B0/B5/V2-L, Swin-T/B/L, CLIP-B32/L14
Detection	7	YOLOv8n/s/m/x, YOLOv11x, RT-DETR-X, Grounding DINO
Segmentation	4	SAM ViT-H, SAM 2, MobileSAM, Mask R-CNN

Audio/Speech (6 models)

Category	Count	Models
Speech-to-Text	3	Parakeet TDT 1.1B, Whisper Large V3, Whisper Turbo
Speaker	3	TitaNet-Large, ECAPA-TDNN, pyannote Diarization

Scientific (4 models)

Category	Count	Models
Protein	3	AlphaFold2, ESM-2 3B, Protenix
Math	1	AlphaGeometry

Medical (2 models)

Category	Count	Models
Imaging	2	nnU-Net, MedSAM

Geospatial (2 models)

Category	Count	Models
Satellite	2	Prithvi 100M, Prithvi 600M

3D/Generative (2 models)

Category	Count	Models
Reconstruction	2	Nerfstudio, 3D Gaussian Splatting

Text Generation (3 models)

Category	Count	Models
LLMs	3	Mistral 7B, Llama 3 70B, Qwen 72B

Total: 38 self-hosted models

API ROUTES SUMMARY

Thermal Management

Method	Path	Description	Permission
GET	/thermal/models	List all models with states	settings:read
GET	/thermal/models/:id	Get model thermal state	settings:read
PUT	/thermal/models/:id	Update thermal state	settings:write
POST	/thermal/bulk	Bulk update states	settings:write

Method	Path	Description	Permission
POST	/thermal/warm/:id	Request model warm-up	(API users)

Mid-Level Services

Method	Path	Description
POST	/perception/detect	Object detection
POST	/perception/segment	Image segmentation
POST	/perception/classify	Image classification
POST	/perception/analyze	Full pipeline
POST	/scientific/protein/embed	Protein embeddings
POST	/scientific/protein/fold	Protein structure
POST	/medical/segment	Medical segmentation
POST	/geospatial/classify	Land use classification
POST	/reconstruction/nerf	3D reconstruction

DEPLOYMENT VERIFICATION

```
# 1. Check thermal states
curl -H "Authorization: Bearer $ADMIN_TOKEN" \
  https://admin-api.YOUR_DOMAIN.com/api/v2/thermal/models

# 2. Update thermal state
curl -X PUT -H "Authorization: Bearer $ADMIN_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"targetState": "WARM"}' \
  https://admin-api.YOUR_DOMAIN.com/api/v2/thermal/models/yolov8m

# 3. Test object detection
curl -X POST -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"imageUrl": "https://YOUR_DOMAIN.com/image.jpg"}' \
  https://api.YOUR_DOMAIN.com/api/v2/perception/detect

# 4. Check service status
curl -H "Authorization: Bearer $TOKEN" \
  https://api.YOUR_DOMAIN.com/api/v2/perception/status
```

NEXT PROMPTS

Continue with: - **Prompt 7**: External Providers & Database Schema/Migrations - **Prompt 8**: Admin Web Dashboard (Next.js) - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 6: Self-Hosted Models & Mid-Level Services RADIANT v2.2.0 - December 2024

â•

END OF SECTION 6

â•

â•

SECTION-07-DATABASE-SCHEMA

SECTION 7: EXTERNAL PROVIDERS & DATABASE SCHEMA (v2.2.0 - CANONICAL)

â•

> **Dependencies:** Sections 0-6 **Creates:** 21 provider integrations, Complete PostgreSQL schema, all migrations

âš ï¸ IMPORTANT: CANONICAL DATABASE SCHEMA

This section contains the ONLY database migration files. Ignore any migration snippets in other sections - use ONLY the migrations defined here.

Canonical Table Names (enforced throughout):

Table	Purpose
`tenants`	Multi-tenant isolation
`users`	End users (not admins)
`administrators`	Platform admins
`invitations`	Admin invitations
`approval_requests`	Two-person approval workflow

Table	Purpose
providers	AI provider configurations
models	AI model definitions
usage_events	Request-level usage tracking
invoices	Billing invoices
audit_logs	Compliance audit trail
revenue_entries	Individual revenue events (subscriptions, credits, AI markup)
cost_entries	Infrastructure costs (AWS, external AI, platform fees)
revenue_daily_aggregates	Pre-computed daily revenue/cost summaries
model_revenue_tracking	Per-model revenue breakdown with markup analysis
accounting_periods	Month-end close tracking
reconciliation_entries	Accounting adjustments
revenue_export_log	Audit trail for accounting exports
preprompt_templates	Reusable pre-prompt patterns with weights
preprompt_instances	Actual pre-prompts used in AGI plans
preprompt_feedback	User feedback with attribution analysis
preprompt_attribution_scores	Learning correlations per factor
preprompt_learning_config	Admin-configurable learning parameters
preprompt_selection_log	Selection reasoning audit trail
user_memory_rules	User personal AI interaction rules
preset_user_rules	Pre-seeded rule templates users can add
user_rule_application_log	Tracks when user rules are applied
memory_categories	Hierarchical categorization of memory types
ai_ethics_standards	Industry AI ethics frameworks (NIST, ISO, EU AI Act)
ai_ethics_principle_standards	Mapping principles to standard sections
provider_rejections	Track provider/model rejections with fallback chain
rejection_patterns	Learn rejection patterns for smarter fallback
user_rejection_notifications	Notify users of rejected requests
model_rejection_stats	Per-model rejection statistics
rejection_analytics	Daily aggregated rejection stats by model/provider/mode
rejection_keyword_stats	Track violation keywords for policy review
rejected_prompt_archive	Archive rejected prompts with full content for analysis

Type Imports

```
import {
  AIProvider,
  AIModel,
  EXTERNAL_PROVIDERS,
  PROVIDER_ENDPOINTS,
  Administrator,
  Invitation,
  ApprovalRequest,
  UsageEvent,
  Invoice,
  AuditLog,
} from '@radiant/shared';
```

RADIANT v2.2.0 - Prompt 7: External Providers & Database Schema/Migrations

Prompt 7 of 9 | Target Size: ~85KB | Version: 3.7.0 | December 2024

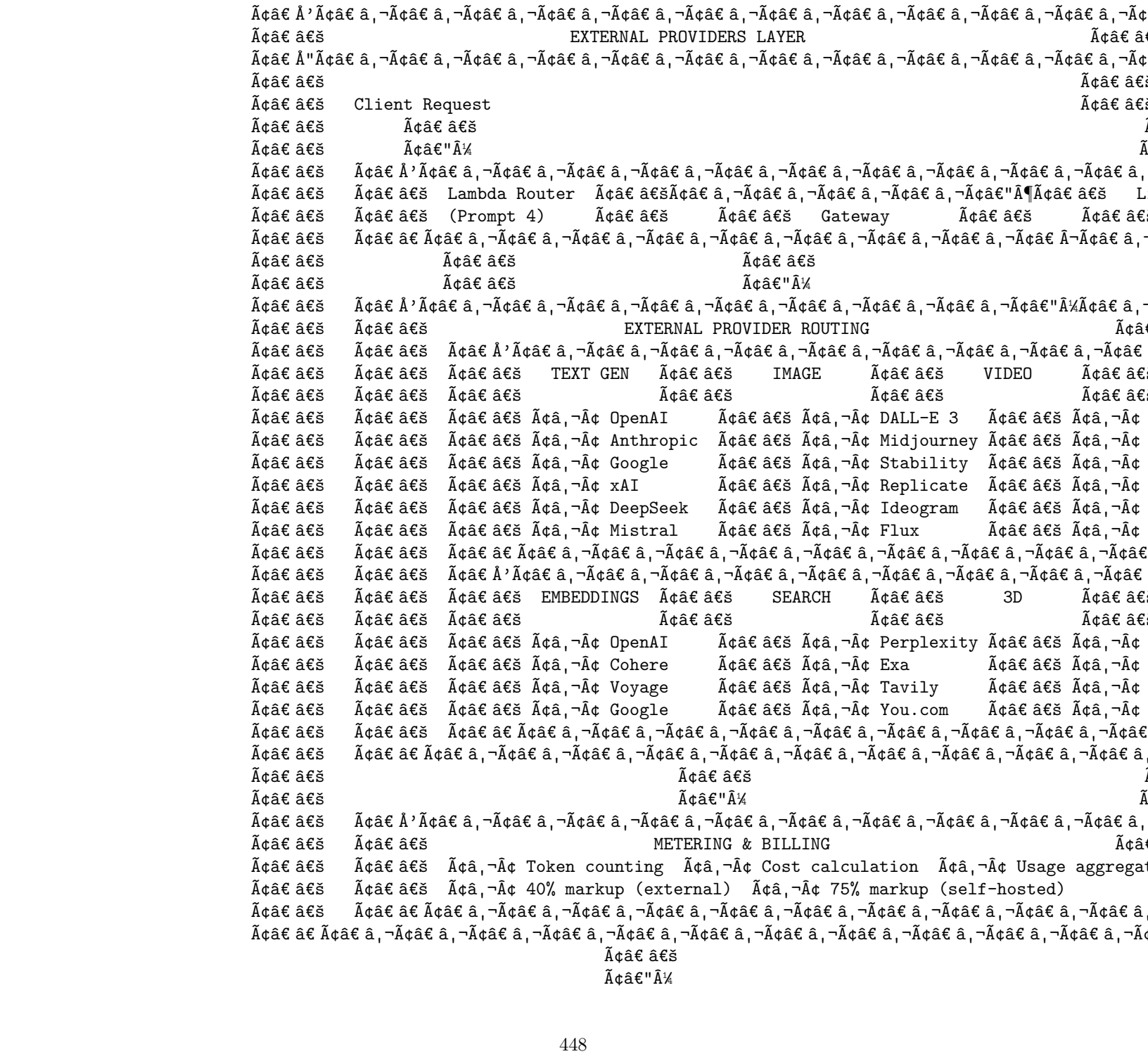
OVERVIEW

This prompt creates the external AI provider integrations and complete database schema:

1. **External Providers** - 21 AI provider configurations with LiteLLM routing
2. **Provider Categories** - Text, Image, Video, Audio, 3D, Embeddings, Search
3. **LiteLLM Configuration** - Complete config.yaml for all external providers
4. **PostgreSQL Schema** - Complete schema with Row-Level Security (RLS)
5. **DynamoDB Tables** - Session, cache, and real-time data tables
6. **Database Migrations** - Versioned migration scripts
7. **Pricing Configuration** - Dynamic pricing with 40% external / 75% self-hosted markup

---

ARCHITECTURE



```
Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
Ã¢â€ â€š                              DATABASE LAYER                              Ã¢â€ â€
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
Ã¢â€ â€š   Ã¢â€ Å’Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
Ã¢â€ â€š   Ã¢â€ â€š    AURORA POSTGRESQL    Ã¢â€ â€š     Ã¢â€ â€š       DYNAMODB          Ã¢â
Ã¢â€ â€š   Ã¢â€ â€š                         Ã¢â€ â€š     Ã¢â€ â€š                         Ã¢â
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Tenants           Ã¢â€ â€š     Ã¢â€ â€š  Ã¢â‚¬Â¢ Sessions
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Users             Ã¢â€ â€š     Ã¢â€ â€š  Ã¢â‚¬Â¢ Chat History
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Admins            Ã¢â€ â€š     Ã¢â€ â€š  Ã¢â‚¬Â¢ Model States
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Providers         Ã¢â€ â€š     Ã¢â€ â€š  Ã¢â‚¬Â¢ Cache
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Models            Ã¢â€ â€š     Ã¢â€ â€š  Ã¢â‚¬Â¢ WebSocket Co
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Usage             Ã¢â€ â€š     Ã¢â€ â€š  Ã¢â‚¬Â¢ Approval Requ
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Billing           Ã¢â€ â€š     Ã¢â€ â€š
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ Audit Logs        Ã¢â€ â€š     Ã¢â€ â€š
Ã¢â€ â€š   Ã¢â€ â€š  Ã¢â‚¬Â¢ RLS Policies      Ã¢â€ â€š     Ã¢â€ â€š  Ã¢â‚¬Â¢ TTL Cleanup
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
```

DIRECTORY STRUCTURE

```
packages/infrastructure/
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ lib/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ config/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ providers/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ index.ts                  # All provider exp
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ text.providers.ts         # Text generation
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ image.providers.ts        # Image generation
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ video.providers.ts        # Video generation
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ audio.providers.ts        # Audio/speech pro
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ embedding.providers.ts    # Embedding provid
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ search.providers.ts       # Search providers
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 3d.providers.ts           # 3D generation pr
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ pricing.config.ts        # Pricing configu
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ litellm/
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ config/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ external.yaml                # External provider conf
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ complete.yaml               # Combined config (exter
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ migrations/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 001_initial_schema.sql          # Base tables
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 002_tenant_isolation.sql        # RLS policies
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 003_ai_models.sql               # Model registry
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 004_usage_billing.sql           # Metering tables
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 005_admin_approval.sql          # Admin workflow tables
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ 006_self_hosted_models.sql      # Self-hosted additions (
```

```
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¬Ã¢â€ â¬ 007_external_providers.sql       # External provider tables
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¬Ã¢â€ â¬ migrations.ts                    # Migration runner
Ã¢â€ â€ Ã¢â€ â¬Ã¢â€ â¬ dynamodb/
    Ã¢â€ â€ Ã¢â€ â¬Ã¢â€ â¬ tables.ts                          # DynamoDB table definitions
```

PART 1: EXTERNAL PROVIDER DEFINITIONS

Provider Pricing Structure

All external providers use a **40% markup** on base costs: - Base cost: Provider's published rate - Billed cost: Base Ãfâ€" 1.40

Self-hosted models (Prompt 6) use a **75% markup** on infrastructure costs.

packages/infrastructure/lib/config/providers/index.ts

```typescript
/**
 * External Provider Registry
 * All external AI providers with pricing and capabilities
 * Pricing includes 40% markup over provider costs
 *
 * RADIANT v2.2.0 - December 2024
 */

export * from './text.providers';
export * from './image.providers';
export * from './video.providers';
export * from './audio.providers';
export * from './embedding.providers';
export * from './search.providers';
export * from './3d.providers';
export * from './pricing.config';

import { TEXT_PROVIDERS } from './text.providers';
import { IMAGE_PROVIDERS } from './image.providers';
import { VIDEO_PROVIDERS } from './video.providers';
import { AUDIO_PROVIDERS } from './audio.providers';
import { EMBEDDING_PROVIDERS } from './embedding.providers';
import { SEARCH_PROVIDERS } from './search.providers';
import { THREE_D_PROVIDERS } from './3d.providers';

// ============================================================================
// PROVIDER TYPES
// ============================================================================
```

```typescript
export type ProviderCategory =
  | 'text_generation'
  | 'image_generation'
  | 'video_generation'
  | 'audio_generation'
  | 'speech_to_text'
  | 'text_to_speech'
  | 'embedding'
  | 'search'
  | '3d_generation'
  | 'reasoning';

export interface ExternalProvider {
  id: string;
  name: string;
  displayName: string;
  category: ProviderCategory;
  description: string;
  website: string;
  apiBaseUrl: string;
  authType: 'api_key' | 'oauth' | 'bearer';
  secretName: string;
  enabled: boolean;
  regions: string[];
  models: ExternalModel[];
  rateLimit?: {
    requestsPerMinute: number;
    tokensPerMinute?: number;
  };
  features: string[];
  compliance?: string[];
}

export interface ExternalModel {
  id: string;
  modelId: string;
  litellmId: string;
  name: string;
  displayName: string;
  description: string;
  category: ProviderCategory;
  capabilities: string[];
  contextWindow?: number;
  maxOutput?: number;
  inputModalities: ('text' | 'image' | 'audio' | 'video')[];
  outputModalities: ('text' | 'image' | 'audio' | 'video' | '3d')[];
```

```
  pricing: ExternalProviderPricing;
  deprecated?: boolean;
  successorModel?: string;
}

export interface ExternalProviderPricing {
  type: 'per_token' | 'per_request' | 'per_second' | 'per_image' | 'per_minute';
  inputCostPer1k?: number;      // $/1K tokens (input)
  outputCostPer1k?: number;     // $/1K tokens (output)
  baseCostPerRequest?: number;  // Base $/request
  costPerSecond?: number;       // $/second (video/audio)
  costPerImage?: number;        // $/image
  costPerMinute?: number;       // $/minute (audio)
  markup: number;               // Multiplier (1.40 for 40%)
  billedInputPer1k?: number;    // Final billed rate
  billedOutputPer1k?: number;   // Final billed rate
}

// ============================================================================
// AGGREGATED REGISTRY
// ============================================================================

export const ALL_EXTERNAL_PROVIDERS: ExternalProvider[] = [
  ...TEXT_PROVIDERS,
  ...IMAGE_PROVIDERS,
  ...VIDEO_PROVIDERS,
  ...AUDIO_PROVIDERS,
  ...EMBEDDING_PROVIDERS,
  ...SEARCH_PROVIDERS,
  ...THREE_D_PROVIDERS,
];

export const PROVIDER_BY_ID = new Map<string, ExternalProvider>(
  ALL_EXTERNAL_PROVIDERS.map(p => [p.id, p])
);

export const ALL_EXTERNAL_MODELS: ExternalModel[] =
  ALL_EXTERNAL_PROVIDERS.flatMap(p => p.models);

export const MODEL_BY_ID = new Map<string, ExternalModel>(
  ALL_EXTERNAL_MODELS.map(m => [m.id, m])
);

export const MODEL_BY_LITELLM_ID = new Map<string, ExternalModel>(
  ALL_EXTERNAL_MODELS.map(m => [m.litellmId, m])
);
```

```typescript
// ============================================================================
// HELPER FUNCTIONS
// ============================================================================

export function calculateCost(
  model: ExternalModel,
  inputTokens: number,
  outputTokens: number,
  additionalUnits?: { images?: number; seconds?: number; minutes?: number }
): { baseCost: number; billedCost: number } {
  const pricing = model.pricing;
  let baseCost = 0;

  if (pricing.type === 'per_token') {
    baseCost =
      (inputTokens / 1000) * (pricing.inputCostPer1k || 0) +
      (outputTokens / 1000) * (pricing.outputCostPer1k || 0);
  } else if (pricing.type === 'per_request') {
    baseCost = pricing.baseCostPerRequest || 0;
  } else if (pricing.type === 'per_image') {
    baseCost = (additionalUnits?.images || 1) * (pricing.costPerImage || 0);
  } else if (pricing.type === 'per_second') {
    baseCost = (additionalUnits?.seconds || 0) * (pricing.costPerSecond || 0);
  } else if (pricing.type === 'per_minute') {
    baseCost = (additionalUnits?.minutes || 0) * (pricing.costPerMinute || 0);
  }

  return {
    baseCost,
    billedCost: baseCost * pricing.markup,
  };
}

export function getProviderSecrets(): Map<string, string> {
  const secrets = new Map<string, string>();
  for (const provider of ALL_EXTERNAL_PROVIDERS) {
    secrets.set(provider.id, provider.secretName);
  }
  return secrets;
}
```

packages/infrastructure/lib/config/providers/text.providers.ts

```typescript
/**
 * Text Generation Providers - RADIANT v4.12
```

```
 * OpenAI (GPT-5, GPT-4.1, O-Series), Anthropic (Claude 4.5), Google (Gemini 3),
 * xAI (Grok 4), DeepSeek (V3), Mistral, Cohere, Perplexity
 *
 * Updated: December 2024
 */

import { ExternalProvider, ExternalModel } from './index';

const MARKUP = 1.40; // 40% markup

// ============================================================================
// OPENAI - GPT-5, GPT-4.1, and O-Series Models
// ============================================================================

const OPENAI_MODELS: ExternalModel[] = [
  // GPT-5 Series (Flagship - December 2024)
  {
    id: 'openai-gpt-5-2',
    modelId: 'gpt-5.2',
    litellmId: 'gpt-5.2',
    name: 'gpt-5.2',
    displayName: 'GPT-5.2',
    description: 'Most capable flagship model with configurable reasoning and 400K context',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'reasoning', 'function_calling', 'json_mode', 'streamin
    contextWindow: 400000,
    maxOutput: 128000,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.00175,
      outputCostPer1k: 0.014,
      cachedInputCostPer1k: 0.000175,
      markup: MARKUP,
      billedInputPer1k: 0.00175 * MARKUP,
      billedOutputPer1k: 0.014 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-12-01',
      family: 'gpt-5',
      version: '5.2',
      isLatest: true,
      supportsBatch: true,
      batchDiscount: 0.5,
    },
```

```
},
{
  id: 'openai-gpt-5-1',
  modelId: 'gpt-5.1',
  litellmId: 'gpt-5.1',
  name: 'gpt-5.1',
  displayName: 'GPT-5.1',
  description: 'Previous flagship with full capabilities and 400K context',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'reasoning', 'function_calling', 'json_mode', 'streamin
  contextWindow: 400000,
  maxOutput: 128000,
  inputModalities: ['text', 'image'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.00125,
    outputCostPer1k: 0.01,
    markup: MARKUP,
    billedInputPer1k: 0.00125 * MARKUP,
    billedOutputPer1k: 0.01 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-10-01',
    family: 'gpt-5',
    version: '5.1',
  },
},
{
  id: 'openai-gpt-5-mini',
  modelId: 'gpt-5-mini',
  litellmId: 'gpt-5-mini',
  name: 'gpt-5-mini',
  displayName: 'GPT-5 Mini',
  description: 'Fast, affordable GPT-5 variant for high-volume tasks',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'function_calling', 'json_mode', 'streaming'],
  contextWindow: 400000,
  maxOutput: 128000,
  inputModalities: ['text', 'image'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.00025,
    outputCostPer1k: 0.002,
    markup: MARKUP,
```

```
    billedInputPer1k: 0.00025 * MARKUP,
    billedOutputPer1k: 0.002 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-11-01',
    family: 'gpt-5',
    version: 'mini',
  },
},
{
  id: 'openai-gpt-5-nano',
  modelId: 'gpt-5-nano',
  litellmId: 'gpt-5-nano',
  name: 'gpt-5-nano',
  displayName: 'GPT-5 Nano',
  description: 'Ultra-fast, most cost-efficient GPT-5 for classification and simple tasks'
  category: 'text_generation',
  capabilities: ['chat', 'function_calling', 'json_mode', 'streaming'],
  contextWindow: 400000,
  maxOutput: 128000,
  inputModalities: ['text'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.00005,
    outputCostPer1k: 0.0004,
    markup: MARKUP,
    billedInputPer1k: 0.00005 * MARKUP,
    billedOutputPer1k: 0.0004 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-11-01',
    family: 'gpt-5',
    version: 'nano',
  },
},
// GPT-4.1 Series (1M Context - April 2024)
{
  id: 'openai-gpt-4-1',
  modelId: 'gpt-4.1-2025-04-14',
  litellmId: 'gpt-4.1-2025-04-14',
  name: 'gpt-4.1',
  displayName: 'GPT-4.1',
  description: 'Best coding model with industry-leading 1M context window',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'function_calling', 'json_mode', 'streaming', 'agents',
```

```
    contextWindow: 1000000,
    maxOutput: 32768,
    inputModalities: ['text', 'image', 'video'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.002,
      outputCostPer1k: 0.008,
      markup: MARKUP,
      billedInputPer1k: 0.002 * MARKUP,
      billedOutputPer1k: 0.008 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-04-14',
      family: 'gpt-4.1',
      version: '4.1',
      isLatest: true,
      supportsFineTuning: true,
    },
  },
  {
    id: 'openai-gpt-4-1-mini',
    modelId: 'gpt-4.1-mini-2025-04-14',
    litellmId: 'gpt-4.1-mini-2025-04-14',
    name: 'gpt-4.1-mini',
    displayName: 'GPT-4.1 Mini',
    description: 'Fast 1M context model beating GPT-4o benchmarks at lower cost',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'function_calling', 'json_mode', 'streaming'],
    contextWindow: 1000000,
    maxOutput: 32768,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0004,
      outputCostPer1k: 0.0016,
      markup: MARKUP,
      billedInputPer1k: 0.0004 * MARKUP,
      billedOutputPer1k: 0.0016 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-04-14',
      family: 'gpt-4.1',
      version: 'mini',
    },
```

```
},
{
  id: 'openai-gpt-4-1-nano',
  modelId: 'gpt-4.1-nano-2025-04-14',
  litellmId: 'gpt-4.1-nano-2025-04-14',
  name: 'gpt-4.1-nano',
  displayName: 'GPT-4.1 Nano',
  description: 'Fastest 1M context model for classification and autocomplete',
  category: 'text_generation',
  capabilities: ['chat', 'function_calling', 'json_mode', 'streaming'],
  contextWindow: 1000000,
  maxOutput: 16384,
  inputModalities: ['text'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.0001,
    outputCostPer1k: 0.0004,
    markup: MARKUP,
    billedInputPer1k: 0.0001 * MARKUP,
    billedOutputPer1k: 0.0004 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-04-14',
    family: 'gpt-4.1',
    version: 'nano',
  },
},
// O-Series Reasoning Models
{
  id: 'openai-o3',
  modelId: 'o3-2025-04-16',
  litellmId: 'o3-2025-04-16',
  name: 'o3',
  displayName: 'O3',
  description: 'Most powerful reasoning model for math, science, and complex coding',
  category: 'reasoning',
  capabilities: ['chat', 'reasoning', 'vision', 'function_calling', 'web_browsing', 'code_
  contextWindow: 200000,
  maxOutput: 100000,
  inputModalities: ['text', 'image'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.002,
    outputCostPer1k: 0.008,
```

```
    markup: MARKUP,
    billedInputPer1k: 0.002 * MARKUP,
    billedOutputPer1k: 0.008 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-04-16',
    family: 'o-series',
    version: 'o3',
    isLatest: true,
    reasoningModel: true,
  },
},
{
  id: 'openai-o3-pro',
  modelId: 'o3-pro-2025-06-10',
  litellmId: 'o3-pro-2025-06-10',
  name: 'o3-pro',
  displayName: 'O3 Pro',
  description: 'Extended compute O3 for maximum reasoning accuracy on hardest problems',
  category: 'reasoning',
  capabilities: ['chat', 'reasoning', 'vision', 'function_calling', 'streaming'],
  contextWindow: 200000,
  maxOutput: 100000,
  inputModalities: ['text', 'image'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.02,
    outputCostPer1k: 0.08,
    markup: MARKUP,
    billedInputPer1k: 0.02 * MARKUP,
    billedOutputPer1k: 0.08 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-06-10',
    family: 'o-series',
    version: 'o3-pro',
    reasoningModel: true,
  },
},
{
  id: 'openai-o4-mini',
  modelId: 'o4-mini-2025-04-16',
  litellmId: 'o4-mini-2025-04-16',
  name: 'o4-mini',
  displayName: 'O4 Mini',
```

```
    description: 'Fast reasoning model, best on AIME math benchmarks',
    category: 'reasoning',
    capabilities: ['chat', 'reasoning', 'vision', 'function_calling', 'streaming'],
    contextWindow: 200000,
    maxOutput: 100000,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0011,
      outputCostPer1k: 0.0044,
      markup: MARKUP,
      billedInputPer1k: 0.0011 * MARKUP,
      billedOutputPer1k: 0.0044 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-04-16',
      family: 'o-series',
      version: 'o4-mini',
      isLatest: true,
      reasoningModel: true,
    },
  },
  {
    id: 'openai-o3-mini',
    modelId: 'o3-mini-2025-01-31',
    litellmId: 'o3-mini-2025-01-31',
    name: 'o3-mini',
    displayName: 'O3 Mini',
    description: 'Small reasoning model optimized for programming and math with configurable
    category: 'reasoning',
    capabilities: ['chat', 'reasoning', 'streaming'],
    contextWindow: 200000,
    maxOutput: 100000,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0011,
      outputCostPer1k: 0.0044,
      markup: MARKUP,
      billedInputPer1k: 0.0011 * MARKUP,
      billedOutputPer1k: 0.0044 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-01-31',
```

```
      family: 'o-series',
      version: 'o3-mini',
      reasoningModel: true,
      reasoningEffort: ['low', 'medium', 'high'],
    },
  },
  // Legacy GPT-4o (still available for audio)
  {
    id: 'openai-gpt-4o',
    modelId: 'gpt-4o-2024-08-06',
    litellmId: 'gpt-4o-2024-08-06',
    name: 'gpt-4o',
    displayName: 'GPT-4o',
    description: 'Multimodal model with native audio input/output capabilities',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'audio', 'function_calling', 'json_mode', 'streaming'],
    contextWindow: 128000,
    maxOutput: 16384,
    inputModalities: ['text', 'image', 'audio'],
    outputModalities: ['text', 'audio'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0025,
      outputCostPer1k: 0.01,
      markup: MARKUP,
      billedInputPer1k: 0.0025 * MARKUP,
      billedOutputPer1k: 0.01 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-08-06',
      family: 'gpt-4o',
      version: '4o',
      legacy: true,
      supportsAudio: true,
    },
  },
];

export const OPENAI_PROVIDER: ExternalProvider = {
  id: 'openai',
  name: 'openai',
  displayName: 'OpenAI',
  category: 'text_generation',
  description: 'Leading AI lab providing GPT-5, GPT-4.1, and O-series reasoning models',
  website: 'https://openai.com',
  apiBaseUrl: 'https://api.openai.com/v1',
```

```
  authType: 'bearer',
  secretName: 'radiant/providers/openai',
  enabled: true,
  regions: ['us-east-1', 'eu-west-1'],
  models: OPENAI_MODELS,
  rateLimit: { requestsPerMinute: 10000, tokensPerMinute: 10000000 },
  features: ['streaming', 'function_calling', 'vision', 'audio', 'json_mode', 'batch_api',
  compliance: ['SOC2', 'GDPR', 'HIPAA'],
  metadata: {
    foundedYear: 2015,
    headquarters: 'San Francisco, CA',
    totalModels: OPENAI_MODELS.length,
    lastUpdated: '2024-12-01',
  },
};

// ============================================================================
// ANTHROPIC - Claude 4.5, 4, and 3.5 Series
// ============================================================================

const ANTHROPIC_MODELS: ExternalModel[] = [
  // Claude 4.5 Series (Latest - November 2024)
  {
    id: 'anthropic-claude-opus-4-5',
    modelId: 'claude-opus-4-5-20251101',
    litellmId: 'anthropic/claude-opus-4-5-20251101',
    name: 'claude-opus-4.5',
    displayName: 'Claude Opus 4.5',
    description: 'Most intelligent model for coding, agents, and computer use',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'tool_use', 'computer_use', 'extended_thinking', 'strea
    contextWindow: 200000,
    maxOutput: 64000,
    inputModalities: ['text', 'image', 'pdf'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.005,
      outputCostPer1k: 0.025,
      cachedInputCostPer1k: 0.0005,
      markup: MARKUP,
      billedInputPer1k: 0.005 * MARKUP,
      billedOutputPer1k: 0.025 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-11-01',
```

```
    family: 'claude-4.5',
    version: 'opus',
    isLatest: true,
    supportsBatch: true,
    batchDiscount: 0.5,
    bedrockId: 'anthropic.claude-opus-4-5-20251101-v1:0',
    vertexId: 'claude-opus-4-5@20251101',
  },
},
{
  id: 'anthropic-claude-sonnet-4-5',
  modelId: 'claude-sonnet-4-5-20250929',
  litellmId: 'anthropic/claude-sonnet-4-5-20250929',
  name: 'claude-sonnet-4.5',
  displayName: 'Claude Sonnet 4.5',
  description: 'Best balance of intelligence, speed, and cost - recommended for most tasks
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'tool_use', 'computer_use', 'extended_thinking', 'strea
  contextWindow: 200000,
  maxOutput: 64000,
  inputModalities: ['text', 'image', 'pdf'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.003,
    outputCostPer1k: 0.015,
    cachedInputCostPer1k: 0.0003,
    longContextInputPer1k: 0.006,
    longContextOutputPer1k: 0.0225,
    markup: MARKUP,
    billedInputPer1k: 0.003 * MARKUP,
    billedOutputPer1k: 0.015 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-09-29',
    family: 'claude-4.5',
    version: 'sonnet',
    isLatest: true,
    isRecommended: true,
    supportsBatch: true,
    batchDiscount: 0.5,
    betaContextWindow: 1000000,
    bedrockId: 'anthropic.claude-sonnet-4-5-20250929-v1:0',
    vertexId: 'claude-sonnet-4-5@20250929',
  },
},
```

```
{
  id: 'anthropic-claude-haiku-4-5',
  modelId: 'claude-haiku-4-5-20251001',
  litellmId: 'anthropic/claude-haiku-4-5-20251001',
  name: 'claude-haiku-4.5',
  displayName: 'Claude Haiku 4.5',
  description: 'Fastest Claude with near-frontier performance for high-volume tasks',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'tool_use', 'streaming', 'batch'],
  contextWindow: 200000,
  maxOutput: 64000,
  inputModalities: ['text', 'image', 'pdf'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.001,
    outputCostPer1k: 0.005,
    cachedInputCostPer1k: 0.0001,
    markup: MARKUP,
    billedInputPer1k: 0.001 * MARKUP,
    billedOutputPer1k: 0.005 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-10-01',
    family: 'claude-4.5',
    version: 'haiku',
    isLatest: true,
    supportsBatch: true,
    batchDiscount: 0.5,
    bedrockId: 'anthropic.claude-haiku-4-5-20251001-v1:0',
    vertexId: 'claude-haiku-4-5@20251001',
  },
},
// Claude 4 Series
{
  id: 'anthropic-claude-opus-4',
  modelId: 'claude-opus-4-20250514',
  litellmId: 'anthropic/claude-opus-4-20250514',
  name: 'claude-opus-4',
  displayName: 'Claude Opus 4',
  description: 'Previous flagship for complex analysis and extended tasks',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'tool_use', 'computer_use', 'extended_thinking', 'strea
  contextWindow: 200000,
  maxOutput: 64000,
  inputModalities: ['text', 'image', 'pdf'],
```

```
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.015,
      outputCostPer1k: 0.075,
      markup: MARKUP,
      billedInputPer1k: 0.015 * MARKUP,
      billedOutputPer1k: 0.075 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-05-14',
      family: 'claude-4',
      version: 'opus',
      bedrockId: 'anthropic.claude-opus-4-20250514-v1:0',
      vertexId: 'claude-opus-4@20250514',
    },
  },
  {
    id: 'anthropic-claude-sonnet-4',
    modelId: 'claude-sonnet-4-20250514',
    litellmId: 'anthropic/claude-sonnet-4-20250514',
    name: 'claude-sonnet-4',
    displayName: 'Claude Sonnet 4',
    description: 'Balanced performance and speed for general use',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'tool_use', 'computer_use', 'extended_thinking', 'strea
    contextWindow: 200000,
    maxOutput: 64000,
    inputModalities: ['text', 'image', 'pdf'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.003,
      outputCostPer1k: 0.015,
      markup: MARKUP,
      billedInputPer1k: 0.003 * MARKUP,
      billedOutputPer1k: 0.015 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-05-14',
      family: 'claude-4',
      version: 'sonnet',
      bedrockId: 'anthropic.claude-sonnet-4-20250514-v1:0',
      vertexId: 'claude-sonnet-4@20250514',
    },
  },
```

```
  // Claude 3.5 Series (Legacy)
  {
    id: 'anthropic-claude-haiku-35',
    modelId: 'claude-3-5-haiku-20241022',
    litellmId: 'anthropic/claude-3-5-haiku-20241022',
    name: 'claude-3.5-haiku',
    displayName: 'Claude 3.5 Haiku',
    description: 'Fast, affordable legacy model for high-volume tasks',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'tool_use', 'streaming'],
    contextWindow: 200000,
    maxOutput: 8192,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0008,
      outputCostPer1k: 0.004,
      markup: MARKUP,
      billedInputPer1k: 0.0008 * MARKUP,
      billedOutputPer1k: 0.004 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-10-22',
      family: 'claude-3.5',
      version: 'haiku',
      legacy: true,
    },
  },
];

export const ANTHROPIC_PROVIDER: ExternalProvider = {
  id: 'anthropic',
  name: 'anthropic',
  displayName: 'Anthropic',
  category: 'text_generation',
  description: 'AI safety company providing Claude 4.5 series - the most intelligent AI assi
  website: 'https://anthropic.com',
  apiBaseUrl: 'https://api.anthropic.com/v1',
  authType: 'api_key',
  secretName: 'radiant/providers/anthropic',
  enabled: true,
  regions: ['us-east-1', 'eu-west-1'],
  models: ANTHROPIC_MODELS,
  rateLimit: { requestsPerMinute: 4000, tokensPerMinute: 400000 },
  features: ['streaming', 'tool_use', 'vision', 'extended_thinking', 'batch_api', 'computer_
```

```
  compliance: ['SOC2', 'GDPR', 'HIPAA'],
  metadata: {
    foundedYear: 2021,
    headquarters: 'San Francisco, CA',
    totalModels: ANTHROPIC_MODELS.length,
    lastUpdated: '2024-11-01',
    cloudPartners: ['AWS Bedrock', 'Google Vertex AI'],
  },
};

// ============================================================================
// GOOGLE - Gemini 3, 2.5, and 2.0 Series
// ============================================================================

const GOOGLE_MODELS: ExternalModel[] = [
  // Gemini 3 Series (Preview - December 2024)
  {
    id: 'google-gemini-3-pro',
    modelId: 'gemini-3-pro-preview',
    litellmId: 'gemini/gemini-3-pro-preview',
    name: 'gemini-3-pro',
    displayName: 'Gemini 3 Pro',
    description: 'Latest flagship with advanced reasoning and native 1M context',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'audio', 'video', 'function_calling', 'thinking', 'grou
    contextWindow: 1000000,
    maxOutput: 65536,
    inputModalities: ['text', 'image', 'audio', 'video', 'pdf'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.002,
      outputCostPer1k: 0.012,
      longContextInputPer1k: 0.004,
      longContextOutputPer1k: 0.018,
      markup: MARKUP,
      billedInputPer1k: 0.002 * MARKUP,
      billedOutputPer1k: 0.012 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-12-01',
      family: 'gemini-3',
      version: 'pro',
      isPreview: true,
      isLatest: true,
    },
```

```
},
{
  id: 'google-gemini-3-flash',
  modelId: 'gemini-3-flash-preview',
  litellmId: 'gemini/gemini-3-flash-preview',
  name: 'gemini-3-flash',
  displayName: 'Gemini 3 Flash',
  description: 'Fast Gemini 3 variant for high-volume multimodal tasks',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'audio', 'video', 'function_calling', 'thinking', 'stre
  contextWindow: 1000000,
  maxOutput: 65536,
  inputModalities: ['text', 'image', 'audio', 'video'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.0005,
    outputCostPer1k: 0.003,
    markup: MARKUP,
    billedInputPer1k: 0.0005 * MARKUP,
    billedOutputPer1k: 0.003 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-12-01',
    family: 'gemini-3',
    version: 'flash',
    isPreview: true,
  },
},
// Gemini 2.5 Series (Stable)
{
  id: 'google-gemini-25-pro',
  modelId: 'gemini-2.5-pro',
  litellmId: 'gemini/gemini-2.5-pro',
  name: 'gemini-2.5-pro',
  displayName: 'Gemini 2.5 Pro',
  description: 'Stable flagship with thinking mode and 1M context',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'audio', 'video', 'function_calling', 'thinking', 'grou
  contextWindow: 1000000,
  maxOutput: 65536,
  inputModalities: ['text', 'image', 'audio', 'video', 'pdf'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.00125,
```

```
      outputCostPer1k: 0.01,
      longContextInputPer1k: 0.0025,
      longContextOutputPer1k: 0.015,
      markup: MARKUP,
      billedInputPer1k: 0.00125 * MARKUP,
      billedOutputPer1k: 0.01 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-09-01',
      family: 'gemini-2.5',
      version: 'pro',
      isLatest: true,
      isRecommended: true,
    },
  },
  {
    id: 'google-gemini-25-flash',
    modelId: 'gemini-2.5-flash',
    litellmId: 'gemini/gemini-2.5-flash',
    name: 'gemini-2.5-flash',
    displayName: 'Gemini 2.5 Flash',
    description: 'Fast multimodal with thinking capabilities',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'audio', 'video', 'function_calling', 'thinking', 'stre
    contextWindow: 1000000,
    maxOutput: 65536,
    inputModalities: ['text', 'image', 'audio', 'video'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0003,
      outputCostPer1k: 0.0025,
      markup: MARKUP,
      billedInputPer1k: 0.0003 * MARKUP,
      billedOutputPer1k: 0.0025 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-09-01',
      family: 'gemini-2.5',
      version: 'flash',
    },
  },
  {
    id: 'google-gemini-25-flash-lite',
    modelId: 'gemini-2.5-flash-lite',
    litellmId: 'gemini/gemini-2.5-flash-lite',
```

```
    name: 'gemini-2.5-flash-lite',
    displayName: 'Gemini 2.5 Flash-Lite',
    description: 'Ultra cost-efficient for high-volume processing',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'function_calling', 'streaming'],
    contextWindow: 1000000,
    maxOutput: 65536,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0001,
      outputCostPer1k: 0.0004,
      markup: MARKUP,
      billedInputPer1k: 0.0001 * MARKUP,
      billedOutputPer1k: 0.0004 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-09-01',
      family: 'gemini-2.5',
      version: 'flash-lite',
    },
  },
  // Gemini 2.0 Series
  {
    id: 'google-gemini-2-flash',
    modelId: 'gemini-2.0-flash-001',
    litellmId: 'gemini/gemini-2.0-flash-001',
    name: 'gemini-2.0-flash',
    displayName: 'Gemini 2.0 Flash',
    description: 'Fast multimodal with real-time capabilities',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'audio', 'video', 'function_calling', 'streaming'],
    contextWindow: 1000000,
    maxOutput: 8192,
    inputModalities: ['text', 'image', 'audio', 'video'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0001,
      outputCostPer1k: 0.0004,
      markup: MARKUP,
      billedInputPer1k: 0.0001 * MARKUP,
      billedOutputPer1k: 0.0004 * MARKUP,
    },
    metadata: {
```

```
      releaseDate: '2024-06-01',
      family: 'gemini-2.0',
      version: 'flash',
      hasFreeVersion: true,
    },
  },
];

export const GOOGLE_PROVIDER: ExternalProvider = {
  id: 'google',
  name: 'google',
  displayName: 'Google AI',
  category: 'text_generation',
  description: 'Google AI providing Gemini 3, 2.5, and 2.0 models with native 1M context',
  website: 'https://ai.google.dev',
  apiBaseUrl: 'https://generativelanguage.googleapis.com/v1beta',
  authType: 'api_key',
  secretName: 'radiant/providers/google',
  enabled: true,
  regions: ['us-east-1', 'eu-west-1', 'ap-northeast-1'],
  models: GOOGLE_MODELS,
  rateLimit: { requestsPerMinute: 1000, tokensPerMinute: 4000000 },
  features: ['streaming', 'function_calling', 'vision', 'audio', 'video', 'thinking', 'groun
  compliance: ['SOC2', 'GDPR', 'HIPAA'],
  metadata: {
    foundedYear: 1998,
    headquarters: 'Mountain View, CA',
    totalModels: GOOGLE_MODELS.length,
    lastUpdated: '2024-12-01',
    cloudPlatform: 'Google Vertex AI',
  },
};

// ============================================================================
// XAI (GROK) - Grok 4 and 3 Series
// ============================================================================

const XAI_MODELS: ExternalModel[] = [
  // Grok 4 Series (Latest)
  {
    id: 'xai-grok-4',
    modelId: 'grok-4-0709',
    litellmId: 'xai/grok-4-0709',
    name: 'grok-4',
    displayName: 'Grok 4',
    description: 'xAI flagship with advanced reasoning capabilities',
```

```
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'reasoning', 'function_calling', 'streaming'],
    contextWindow: 256000,
    maxOutput: 256000,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.003,
      outputCostPer1k: 0.015,
      markup: MARKUP,
      billedInputPer1k: 0.003 * MARKUP,
      billedOutputPer1k: 0.015 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-07-09',
      family: 'grok-4',
      version: '4',
      isLatest: true,
    },
  },
  {
    id: 'xai-grok-4-1-fast',
    modelId: 'grok-4-1-fast-reasoning',
    litellmId: 'xai/grok-4-1-fast-reasoning',
    name: 'grok-4.1-fast',
    displayName: 'Grok 4.1 Fast',
    description: 'Best value: near-flagship performance at 1/15th price with 2M context',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'reasoning', 'function_calling', 'streaming', 'tool_cal
    contextWindow: 2000000,
    maxOutput: 2000000,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0002,
      outputCostPer1k: 0.0005,
      markup: MARKUP,
      billedInputPer1k: 0.0002 * MARKUP,
      billedOutputPer1k: 0.0005 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-11-01',
      family: 'grok-4.1',
      version: 'fast-reasoning',
```

```
    isLatest: true,
    isRecommended: true,
    bestValue: true,
  },
},
{
  id: 'xai-grok-4-fast',
  modelId: 'grok-4-fast-reasoning',
  litellmId: 'xai/grok-4-fast-reasoning',
  name: 'grok-4-fast',
  displayName: 'Grok 4 Fast Reasoning',
  description: 'Fast reasoning with massive 2M context window',
  category: 'reasoning',
  capabilities: ['chat', 'vision', 'reasoning', 'function_calling', 'streaming'],
  contextWindow: 2000000,
  maxOutput: 2000000,
  inputModalities: ['text', 'image'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.0002,
    outputCostPer1k: 0.0005,
    markup: MARKUP,
    billedInputPer1k: 0.0002 * MARKUP,
    billedOutputPer1k: 0.0005 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-09-01',
    family: 'grok-4',
    version: 'fast-reasoning',
  },
},
// Grok 3 Series
{
  id: 'xai-grok-3',
  modelId: 'grok-3',
  litellmId: 'xai/grok-3',
  name: 'grok-3',
  displayName: 'Grok 3',
  description: 'Previous flagship with real-time X/Twitter knowledge',
  category: 'text_generation',
  capabilities: ['chat', 'vision', 'function_calling', 'streaming'],
  contextWindow: 131072,
  maxOutput: 131072,
  inputModalities: ['text', 'image'],
  outputModalities: ['text'],
```

```
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.003,
      outputCostPer1k: 0.015,
      markup: MARKUP,
      billedInputPer1k: 0.003 * MARKUP,
      billedOutputPer1k: 0.015 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-03-01',
      family: 'grok-3',
      version: '3',
    },
  },
  {
    id: 'xai-grok-3-mini',
    modelId: 'grok-3-mini',
    litellmId: 'xai/grok-3-mini',
    name: 'grok-3-mini',
    displayName: 'Grok 3 Mini',
    description: 'Smaller, faster Grok for everyday tasks',
    category: 'text_generation',
    capabilities: ['chat', 'function_calling', 'streaming'],
    contextWindow: 131072,
    maxOutput: 16384,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0003,
      outputCostPer1k: 0.0005,
      markup: MARKUP,
      billedInputPer1k: 0.0003 * MARKUP,
      billedOutputPer1k: 0.0005 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-03-01',
      family: 'grok-3',
      version: 'mini',
    },
  },
  // Grok 2 Series
  {
    id: 'xai-grok-2',
    modelId: 'grok-2-1212',
    litellmId: 'xai/grok-2-1212',
```

```
    name: 'grok-2',
    displayName: 'Grok 2',
    description: 'Stable previous generation model',
    category: 'text_generation',
    capabilities: ['chat', 'vision', 'function_calling', 'streaming'],
    contextWindow: 131072,
    maxOutput: 131072,
    inputModalities: ['text', 'image'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.002,
      outputCostPer1k: 0.01,
      markup: MARKUP,
      billedInputPer1k: 0.002 * MARKUP,
      billedOutputPer1k: 0.01 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-12-12',
      family: 'grok-2',
      version: '2',
      legacy: true,
    },
  },
];

export const XAI_PROVIDER: ExternalProvider = {
  id: 'xai',
  name: 'xai',
  displayName: 'xAI (Grok)',
  category: 'text_generation',
  description: 'xAI providing Grok 4 series with industry-leading 2M context and competitive
  website: 'https://x.ai',
  apiBaseUrl: 'https://api.x.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/xai',
  enabled: true,
  regions: ['us-east-1'],
  models: XAI_MODELS,
  rateLimit: { requestsPerMinute: 1000, tokensPerMinute: 1000000 },
  features: ['streaming', 'function_calling', 'vision', 'reasoning', 'web_search', 'x_search
  compliance: ['SOC2'],
  metadata: {
    foundedYear: 2023,
    headquarters: 'San Francisco, CA',
    totalModels: XAI_MODELS.length,
```

```
    lastUpdated: '2024-11-01',
    serverTools: ['web_search', 'x_search', 'code_execution', 'document_search'],
    toolCallCost: 0.005,
  },
};

// ============================================================================
// DEEPSEEK - V3 Models (Best Value in Market)
// ============================================================================

const DEEPSEEK_MODELS: ExternalModel[] = [
  {
    id: 'deepseek-chat',
    modelId: 'deepseek-chat',
    litellmId: 'deepseek/deepseek-chat',
    name: 'deepseek-chat',
    displayName: 'DeepSeek Chat (V3.2)',
    description: 'Best value in market: frontier performance at 90% lower cost',
    category: 'text_generation',
    capabilities: ['chat', 'function_calling', 'json_mode', 'streaming', 'prefix_completion
    contextWindow: 128000,
    maxOutput: 8192,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.00028,
      outputCostPer1k: 0.00042,
      cachedInputCostPer1k: 0.000028,
      markup: MARKUP,
      billedInputPer1k: 0.00028 * MARKUP,
      billedOutputPer1k: 0.00042 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-12-01',
      family: 'deepseek-v3',
      version: '3.2',
      isLatest: true,
      isRecommended: true,
      bestValue: true,
      openAICompatible: true,
    },
  },
  {
    id: 'deepseek-reasoner',
    modelId: 'deepseek-reasoner',
```

```
    litellmId: 'deepseek/deepseek-reasoner',
    name: 'deepseek-reasoner',
    displayName: 'DeepSeek R1',
    description: 'Advanced reasoning model rivaling O1 at fraction of cost',
    category: 'reasoning',
    capabilities: ['chat', 'reasoning', 'function_calling', 'streaming', 'thinking_in_tool_u
    contextWindow: 128000,
    maxOutput: 64000,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.00028,
      outputCostPer1k: 0.00042,
      cachedInputCostPer1k: 0.000028,
      markup: MARKUP,
      billedInputPer1k: 0.00028 * MARKUP,
      billedOutputPer1k: 0.00042 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-12-01',
      family: 'deepseek-r1',
      version: 'r1',
      isLatest: true,
      reasoningModel: true,
      openAICompatible: true,
    },
  },
];

export const DEEPSEEK_PROVIDER: ExternalProvider = {
  id: 'deepseek',
  name: 'deepseek',
  displayName: 'DeepSeek',
  category: 'text_generation',
  description: 'Most cost-effective frontier AI: 90% cheaper than competitors with comparabl
  website: 'https://deepseek.com',
  apiBaseUrl: 'https://api.deepseek.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/deepseek',
  enabled: true,
  regions: ['global'],
  models: DEEPSEEK_MODELS,
  rateLimit: { requestsPerMinute: 1000 },
  features: ['streaming', 'function_calling', 'reasoning', 'json_mode', 'context_caching',
  compliance: [],
```

```
  metadata: {
    foundedYear: 2023,
    headquarters: 'Hangzhou, China',
    totalModels: DEEPSEEK_MODELS.length,
    lastUpdated: '2024-12-01',
    openAISDKCompatible: true,
    bestValueProvider: true,
  },
};

// ============================================================================
// MISTRAL - European Frontier Models
// ============================================================================

const MISTRAL_MODELS: ExternalModel[] = [
  {
    id: 'mistral-large',
    modelId: 'mistral-large-2411',
    litellmId: 'mistral/mistral-large-2411',
    name: 'mistral-large',
    displayName: 'Mistral Large 2.1',
    description: 'European flagship for complex reasoning and enterprise tasks',
    category: 'text_generation',
    capabilities: ['chat', 'function_calling', 'json_mode', 'streaming'],
    contextWindow: 128000,
    maxOutput: 4096,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.002,
      outputCostPer1k: 0.006,
      markup: MARKUP,
      billedInputPer1k: 0.002 * MARKUP,
      billedOutputPer1k: 0.006 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-11-01',
      family: 'mistral-large',
      version: '2.1',
      isLatest: true,
    },
  },
  {
    id: 'mistral-medium',
    modelId: 'mistral-medium-2505',
```

```
    litellmId: 'mistral/mistral-medium-2505',
    name: 'mistral-medium',
    displayName: 'Mistral Medium 3',
    description: 'Balanced European model for general enterprise tasks',
    category: 'text_generation',
    capabilities: ['chat', 'function_calling', 'json_mode', 'streaming'],
    contextWindow: 128000,
    maxOutput: 4096,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0004,
      outputCostPer1k: 0.002,
      markup: MARKUP,
      billedInputPer1k: 0.0004 * MARKUP,
      billedOutputPer1k: 0.002 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-05-01',
      family: 'mistral-medium',
      version: '3',
    },
  },
  {
    id: 'mistral-small',
    modelId: 'mistral-small-2409',
    litellmId: 'mistral/mistral-small-2409',
    name: 'mistral-small',
    displayName: 'Mistral Small 2',
    description: 'Cost-efficient for everyday tasks',
    category: 'text_generation',
    capabilities: ['chat', 'function_calling', 'json_mode', 'streaming'],
    contextWindow: 128000,
    maxOutput: 4096,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0001,
      outputCostPer1k: 0.0003,
      markup: MARKUP,
      billedInputPer1k: 0.0001 * MARKUP,
      billedOutputPer1k: 0.0003 * MARKUP,
    },
    metadata: {
```

```
    releaseDate: '2024-09-01',
    family: 'mistral-small',
    version: '2',
  },
},
{
  id: 'mistral-pixtral-large',
  modelId: 'pixtral-large-2411',
  litellmId: 'mistral/pixtral-large-2411',
  name: 'pixtral-large',
  displayName: 'Pixtral Large',
  description: 'Vision model for OCR, document analysis, and image understanding',
  category: 'vision',
  capabilities: ['chat', 'vision', 'function_calling', 'streaming'],
  contextWindow: 128000,
  maxOutput: 4096,
  inputModalities: ['text', 'image'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.002,
    outputCostPer1k: 0.006,
    markup: MARKUP,
    billedInputPer1k: 0.002 * MARKUP,
    billedOutputPer1k: 0.006 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-11-01',
    family: 'pixtral',
    version: 'large',
  },
},
{
  id: 'mistral-codestral',
  modelId: 'codestral-2501',
  litellmId: 'mistral/codestral-2501',
  name: 'codestral',
  displayName: 'Codestral',
  description: 'Specialized code model supporting 80+ languages with fill-in-the-middle',
  category: 'code',
  capabilities: ['chat', 'code_completion', 'fim', 'streaming'],
  contextWindow: 256000,
  maxOutput: 8192,
  inputModalities: ['text'],
  outputModalities: ['text'],
  pricing: {
```

```
      type: 'per_token',
      inputCostPer1k: 0.0002,
      outputCostPer1k: 0.0006,
      markup: MARKUP,
      billedInputPer1k: 0.0002 * MARKUP,
      billedOutputPer1k: 0.0006 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-01-01',
      family: 'codestral',
      version: '2501',
      supportedLanguages: 80,
    },
  },
  {
    id: 'mistral-magistral-medium',
    modelId: 'magistral-medium-2509',
    litellmId: 'mistral/magistral-medium-2509',
    name: 'magistral-medium',
    displayName: 'Magistral Medium',
    description: 'Reasoning model with configurable thinking effort levels',
    category: 'reasoning',
    capabilities: ['chat', 'reasoning', 'function_calling', 'streaming'],
    contextWindow: 128000,
    maxOutput: 8192,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.001,
      outputCostPer1k: 0.003,
      markup: MARKUP,
      billedInputPer1k: 0.001 * MARKUP,
      billedOutputPer1k: 0.003 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-09-01',
      family: 'magistral',
      version: 'medium',
      reasoningModel: true,
      reasoningEffort: ['low', 'medium', 'high'],
    },
  },
];

export const MISTRAL_PROVIDER: ExternalProvider = {
```

```
  id: 'mistral',
  name: 'mistral',
  displayName: 'Mistral AI',
  category: 'text_generation',
  description: 'European frontier models with GDPR compliance and specialized coding/vision
  website: 'https://mistral.ai',
  apiBaseUrl: 'https://api.mistral.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/mistral',
  enabled: true,
  regions: ['eu-west-1', 'us-east-1'],
  models: MISTRAL_MODELS,
  rateLimit: { requestsPerMinute: 1000 },
  features: ['streaming', 'function_calling', 'vision', 'code_completion', 'fim', 'reasoning
  compliance: ['GDPR', 'SOC2'],
  metadata: {
    foundedYear: 2023,
    headquarters: 'Paris, France',
    totalModels: MISTRAL_MODELS.length,
    lastUpdated: '2024-11-01',
    europeanProvider: true,
  },
};

// ============================================================================
// COHERE - Enterprise RAG Models
// ============================================================================

const COHERE_MODELS: ExternalModel[] = [
  {
    id: 'cohere-command-r-plus',
    modelId: 'command-r-plus-08-2024',
    litellmId: 'cohere/command-r-plus-08-2024',
    name: 'command-r-plus',
    displayName: 'Command R+',
    description: 'Enterprise model optimized for RAG and complex tool use',
    category: 'text_generation',
    capabilities: ['chat', 'rag', 'function_calling', 'streaming'],
    contextWindow: 128000,
    maxOutput: 4096,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0025,
      outputCostPer1k: 0.01,
```

```
    markup: MARKUP,
    billedInputPer1k: 0.0025 * MARKUP,
    billedOutputPer1k: 0.01 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-08-01',
    family: 'command-r',
    version: 'plus',
    isLatest: true,
  },
},
{
  id: 'cohere-command-r',
  modelId: 'command-r-08-2024',
  litellmId: 'cohere/command-r-08-2024',
  name: 'command-r',
  displayName: 'Command R',
  description: 'Balanced model for general enterprise tasks',
  category: 'text_generation',
  capabilities: ['chat', 'rag', 'function_calling', 'streaming'],
  contextWindow: 128000,
  maxOutput: 4096,
  inputModalities: ['text'],
  outputModalities: ['text'],
  pricing: {
    type: 'per_token',
    inputCostPer1k: 0.00015,
    outputCostPer1k: 0.0006,
    markup: MARKUP,
    billedInputPer1k: 0.00015 * MARKUP,
    billedOutputPer1k: 0.0006 * MARKUP,
  },
  metadata: {
    releaseDate: '2024-08-01',
    family: 'command-r',
    version: 'base',
  },
},
{
  id: 'cohere-embed-v4',
  modelId: 'embed-v4.0',
  litellmId: 'cohere/embed-v4.0',
  name: 'embed-v4',
  displayName: 'Embed V4',
  description: 'State-of-the-art embeddings for semantic search and RAG',
  category: 'embeddings',
```

```
    capabilities: ['embeddings'],
    contextWindow: 128000,
    maxOutput: 0,
    inputModalities: ['text'],
    outputModalities: ['embeddings'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.0001,
      outputCostPer1k: 0,
      markup: MARKUP,
      billedInputPer1k: 0.0001 * MARKUP,
      billedOutputPer1k: 0,
    },
    metadata: {
      releaseDate: '2024-10-01',
      family: 'embed',
      version: 'v4',
      dimensions: 1024,
    },
  },
];

export const COHERE_PROVIDER: ExternalProvider = {
  id: 'cohere',
  name: 'cohere',
  displayName: 'Cohere',
  category: 'text_generation',
  description: 'Enterprise AI specialized in RAG and semantic search',
  website: 'https://cohere.com',
  apiBaseUrl: 'https://api.cohere.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/cohere',
  enabled: true,
  regions: ['us-east-1', 'eu-west-1'],
  models: COHERE_MODELS,
  rateLimit: { requestsPerMinute: 1000 },
  features: ['streaming', 'function_calling', 'rag', 'embeddings', 'rerank'],
  compliance: ['SOC2', 'GDPR'],
  metadata: {
    foundedYear: 2019,
    headquarters: 'Toronto, Canada',
    totalModels: COHERE_MODELS.length,
    lastUpdated: '2024-10-01',
    enterpriseFocus: true,
  },
};
```

```typescript
// ============================================================================
// PERPLEXITY - Search-Augmented Models
// ============================================================================

const PERPLEXITY_MODELS: ExternalModel[] = [
  {
    id: 'perplexity-sonar-huge',
    modelId: 'sonar-huge-online',
    litellmId: 'perplexity/sonar-huge-online',
    name: 'sonar-huge',
    displayName: 'Sonar Huge',
    description: 'Most capable search-augmented model with real-time web access',
    category: 'search',
    capabilities: ['chat', 'search', 'citations', 'streaming'],
    contextWindow: 200000,
    maxOutput: 8192,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.005,
      outputCostPer1k: 0.005,
      searchCostPerRequest: 0.005,
      markup: MARKUP,
      billedInputPer1k: 0.005 * MARKUP,
      billedOutputPer1k: 0.005 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-09-01',
      family: 'sonar',
      version: 'huge',
      isLatest: true,
      searchEnabled: true,
    },
  },
  {
    id: 'perplexity-sonar-pro',
    modelId: 'sonar-pro-online',
    litellmId: 'perplexity/sonar-pro-online',
    name: 'sonar-pro',
    displayName: 'Sonar Pro',
    description: 'Balanced search model for general research and fact-checking',
    category: 'search',
    capabilities: ['chat', 'search', 'citations', 'streaming'],
    contextWindow: 200000,
```

```
    maxOutput: 8192,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.003,
      outputCostPer1k: 0.015,
      searchCostPerRequest: 0.005,
      markup: MARKUP,
      billedInputPer1k: 0.003 * MARKUP,
      billedOutputPer1k: 0.015 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-09-01',
      family: 'sonar',
      version: 'pro',
      searchEnabled: true,
    },
  },
  {
    id: 'perplexity-sonar',
    modelId: 'sonar-online',
    litellmId: 'perplexity/sonar-online',
    name: 'sonar',
    displayName: 'Sonar',
    description: 'Fast search-augmented model for quick lookups',
    category: 'search',
    capabilities: ['chat', 'search', 'citations', 'streaming'],
    contextWindow: 127000,
    maxOutput: 8192,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: {
      type: 'per_token',
      inputCostPer1k: 0.001,
      outputCostPer1k: 0.001,
      searchCostPerRequest: 0.005,
      markup: MARKUP,
      billedInputPer1k: 0.001 * MARKUP,
      billedOutputPer1k: 0.001 * MARKUP,
    },
    metadata: {
      releaseDate: '2024-09-01',
      family: 'sonar',
      version: 'base',
      searchEnabled: true,
```

```
    },
  },
];

export const PERPLEXITY_PROVIDER: ExternalProvider = {
  id: 'perplexity',
  name: 'perplexity',
  displayName: 'Perplexity',
  category: 'search',
  description: 'Search-augmented AI with real-time web access and automatic citations',
  website: 'https://perplexity.ai',
  apiBaseUrl: 'https://api.perplexity.ai',
  authType: 'bearer',
  secretName: 'radiant/providers/perplexity',
  enabled: true,
  regions: ['us-east-1'],
  models: PERPLEXITY_MODELS,
  rateLimit: { requestsPerMinute: 100 },
  features: ['streaming', 'search', 'citations', 'web_access'],
  compliance: ['SOC2'],
  metadata: {
    foundedYear: 2022,
    headquarters: 'San Francisco, CA',
    totalModels: PERPLEXITY_MODELS.length,
    lastUpdated: '2024-09-01',
    searchProvider: true,
  },
};

// ============================================================================
// EXPORT ALL TEXT PROVIDERS
// ============================================================================

export const TEXT_PROVIDERS: ExternalProvider[] = [
  OPENAI_PROVIDER,
  ANTHROPIC_PROVIDER,
  GOOGLE_PROVIDER,
  XAI_PROVIDER,
  DEEPSEEK_PROVIDER,
  MISTRAL_PROVIDER,
  COHERE_PROVIDER,
  PERPLEXITY_PROVIDER,
];

/**
 * Image Generation Providers
```

```
 * DALL-E, Stability, Flux, Ideogram, Midjourney
 */

import { ExternalProvider, ExternalModel } from './index';

const MARKUP = 1.40;

// ============================================================================
// OPENAI (DALL-E)
// ============================================================================

const DALLE_MODELS: ExternalModel[] = [
  {
    id: 'openai-dall-e-3',
    modelId: 'dall-e-3',
    litellmId: 'dall-e-3',
    name: 'dall-e-3',
    displayName: 'DALL-E 3',
    description: 'Latest image generation with superior quality',
    category: 'image_generation',
    capabilities: ['text_to_image', 'hd', 'wide_format'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.04, markup: MARKUP },
  },
  {
    id: 'openai-dall-e-3-hd',
    modelId: 'dall-e-3-hd',
    litellmId: 'dall-e-3',
    name: 'dall-e-3-hd',
    displayName: 'DALL-E 3 HD',
    description: 'High-definition DALL-E 3 images',
    category: 'image_generation',
    capabilities: ['text_to_image', 'hd', 'wide_format'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.08, markup: MARKUP },
  },
];

export const DALLE_PROVIDER: ExternalProvider = {
  id: 'openai-images',
  name: 'openai-images',
  displayName: 'OpenAI Images',
  category: 'image_generation',
  description: 'OpenAI DALL-E image generation',
```

```
  website: 'https://openai.com',
  apiBaseUrl: 'https://api.openai.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/openai',
  enabled: true,
  regions: ['us-east-1'],
  models: DALLE_MODELS,
  features: ['text_to_image', 'image_editing', 'variations'],
};

// ============================================================================
// STABILITY AI
// ============================================================================

const STABILITY_MODELS: ExternalModel[] = [
  {
    id: 'stability-sdxl',
    modelId: 'stable-diffusion-xl-1024-v1-0',
    litellmId: 'stability/stable-diffusion-xl-1024-v1-0',
    name: 'sdxl',
    displayName: 'Stable Diffusion XL',
    description: 'High-quality 1024px image generation',
    category: 'image_generation',
    capabilities: ['text_to_image', 'image_to_image'],
    inputModalities: ['text', 'image'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.002, markup: MARKUP },
  },
  {
    id: 'stability-sd3-large',
    modelId: 'sd3-large',
    litellmId: 'stability/sd3-large',
    name: 'sd3-large',
    displayName: 'Stable Diffusion 3 Large',
    description: 'Latest SD3 with improved quality',
    category: 'image_generation',
    capabilities: ['text_to_image'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.065, markup: MARKUP },
  },
  {
    id: 'stability-ultra',
    modelId: 'stable-image-ultra',
    litellmId: 'stability/stable-image-ultra',
    name: 'stable-image-ultra',
```

```
    displayName: 'Stable Image Ultra',
    description: 'Premium image generation with maximum quality',
    category: 'image_generation',
    capabilities: ['text_to_image'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.08, markup: MARKUP },
  },
];

export const STABILITY_PROVIDER: ExternalProvider = {
  id: 'stability',
  name: 'stability',
  displayName: 'Stability AI',
  category: 'image_generation',
  description: 'Open-source focused image generation',
  website: 'https://stability.ai',
  apiBaseUrl: 'https://api.stability.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/stability',
  enabled: true,
  regions: ['us-east-1'],
  models: STABILITY_MODELS,
  features: ['text_to_image', 'image_to_image', 'inpainting', 'upscaling'],
};

// ============================================================================
// FLUX (BLACK FOREST LABS)
// ============================================================================

const FLUX_MODELS: ExternalModel[] = [
  {
    id: 'flux-pro-11',
    modelId: 'flux-pro-1.1',
    litellmId: 'replicate/black-forest-labs/flux-1.1-pro',
    name: 'flux-pro-1.1',
    displayName: 'FLUX 1.1 Pro',
    description: 'State-of-the-art text-to-image with 6x faster generation',
    category: 'image_generation',
    capabilities: ['text_to_image'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.04, markup: MARKUP },
  },
  {
    id: 'flux-pro-ultra',
```

```
    modelId: 'flux-pro-1.1-ultra',
    litellmId: 'replicate/black-forest-labs/flux-1.1-pro-ultra',
    name: 'flux-pro-1.1-ultra',
    displayName: 'FLUX 1.1 Pro Ultra',
    description: 'Ultra-high resolution 4MP images',
    category: 'image_generation',
    capabilities: ['text_to_image', '4mp'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.06, markup: MARKUP },
  },
  {
    id: 'flux-schnell',
    modelId: 'flux-schnell',
    litellmId: 'replicate/black-forest-labs/flux-schnell',
    name: 'flux-schnell',
    displayName: 'FLUX Schnell',
    description: 'Fast, efficient image generation',
    category: 'image_generation',
    capabilities: ['text_to_image', 'fast'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.003, markup: MARKUP },
  },
];

export const FLUX_PROVIDER: ExternalProvider = {
  id: 'flux',
  name: 'flux',
  displayName: 'FLUX (Black Forest Labs)',
  category: 'image_generation',
  description: 'State-of-the-art open image generation',
  website: 'https://blackforestlabs.ai',
  apiBaseUrl: 'https://api.replicate.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/replicate',
  enabled: true,
  regions: ['us-east-1'],
  models: FLUX_MODELS,
  features: ['text_to_image', 'fast_generation', 'high_resolution'],
};

// ============================================================================
// IDEOGRAM
// ============================================================================
```

```typescript
const IDEOGRAM_MODELS: ExternalModel[] = [
  {
    id: 'ideogram-v2',
    modelId: 'ideogram-v2',
    litellmId: 'ideogram/ideogram-v2',
    name: 'ideogram-v2',
    displayName: 'Ideogram V2',
    description: 'Advanced text rendering in images',
    category: 'image_generation',
    capabilities: ['text_to_image', 'text_rendering'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.08, markup: MARKUP },
  },
  {
    id: 'ideogram-v2-turbo',
    modelId: 'ideogram-v2-turbo',
    litellmId: 'ideogram/ideogram-v2-turbo',
    name: 'ideogram-v2-turbo',
    displayName: 'Ideogram V2 Turbo',
    description: 'Fast with good text rendering',
    category: 'image_generation',
    capabilities: ['text_to_image', 'text_rendering', 'fast'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.05, markup: MARKUP },
  },
];

export const IDEOGRAM_PROVIDER: ExternalProvider = {
  id: 'ideogram',
  name: 'ideogram',
  displayName: 'Ideogram',
  category: 'image_generation',
  description: 'Specialized in accurate text rendering',
  website: 'https://ideogram.ai',
  apiBaseUrl: 'https://api.ideogram.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/ideogram',
  enabled: true,
  regions: ['us-east-1'],
  models: IDEOGRAM_MODELS,
  features: ['text_to_image', 'text_rendering', 'logos'],
};

// ============================================================================
```

```typescript
// MIDJOURNEY
// ============================================================================

const MIDJOURNEY_MODELS: ExternalModel[] = [
  {
    id: 'midjourney-v6',
    modelId: 'midjourney-v6',
    litellmId: 'midjourney/v6',
    name: 'midjourney-v6',
    displayName: 'Midjourney V6',
    description: 'Premium artistic image generation',
    category: 'image_generation',
    capabilities: ['text_to_image', 'artistic'],
    inputModalities: ['text'],
    outputModalities: ['image'],
    pricing: { type: 'per_image', costPerImage: 0.10, markup: MARKUP },
  },
];

export const MIDJOURNEY_PROVIDER: ExternalProvider = {
  id: 'midjourney',
  name: 'midjourney',
  displayName: 'Midjourney',
  category: 'image_generation',
  description: 'Premium artistic image generation',
  website: 'https://midjourney.com',
  apiBaseUrl: 'https://api.mymidjourney.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/midjourney',
  enabled: true,
  regions: ['us-east-1'],
  models: MIDJOURNEY_MODELS,
  features: ['text_to_image', 'artistic', 'premium_quality'],
};

export const IMAGE_PROVIDERS: ExternalProvider[] = [
  DALLE_PROVIDER,
  STABILITY_PROVIDER,
  FLUX_PROVIDER,
  IDEOGRAM_PROVIDER,
  MIDJOURNEY_PROVIDER,
];
```

packages/infrastructure/lib/config/providers/video.providers.ts

```
/**
 * Video Generation Providers
 * Runway, Luma, Pika, Kling, HailuoAI, Veo
 */

import { ExternalProvider, ExternalModel } from './index';

const MARKUP = 1.40;

// ============================================================================
// RUNWAY
// ============================================================================

const RUNWAY_MODELS: ExternalModel[] = [
  {
    id: 'runway-gen3-alpha-turbo',
    modelId: 'gen3a_turbo',
    litellmId: 'runway/gen3a_turbo',
    name: 'gen3-alpha-turbo',
    displayName: 'Gen-3 Alpha Turbo',
    description: 'Fast, high-quality video generation',
    category: 'video_generation',
    capabilities: ['text_to_video', 'image_to_video'],
    inputModalities: ['text', 'image'],
    outputModalities: ['video'],
    pricing: { type: 'per_second', costPerSecond: 0.05, markup: MARKUP },
  },
  {
    id: 'runway-gen3-alpha',
    modelId: 'gen3a',
    litellmId: 'runway/gen3a',
    name: 'gen3-alpha',
    displayName: 'Gen-3 Alpha',
    description: 'Maximum quality video generation',
    category: 'video_generation',
    capabilities: ['text_to_video', 'image_to_video'],
    inputModalities: ['text', 'image'],
    outputModalities: ['video'],
    pricing: { type: 'per_second', costPerSecond: 0.10, markup: MARKUP },
  },
];

export const RUNWAY_PROVIDER: ExternalProvider = {
  id: 'runway',
```

```
  name: 'runway',
  displayName: 'Runway',
  category: 'video_generation',
  description: 'Professional AI video generation tools',
  website: 'https://runway.ml',
  apiBaseUrl: 'https://api.runway.ml/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/runway',
  enabled: true,
  regions: ['us-east-1'],
  models: RUNWAY_MODELS,
  features: ['text_to_video', 'image_to_video', 'motion_brush'],
};

// ============================================================================
// LUMA AI
// ============================================================================

const LUMA_MODELS: ExternalModel[] = [
  {
    id: 'luma-dream-machine',
    modelId: 'dream-machine',
    litellmId: 'luma/dream-machine',
    name: 'dream-machine',
    displayName: 'Dream Machine',
    description: 'Fast, cinematic video generation',
    category: 'video_generation',
    capabilities: ['text_to_video', 'image_to_video'],
    inputModalities: ['text', 'image'],
    outputModalities: ['video'],
    pricing: { type: 'per_second', costPerSecond: 0.032, markup: MARKUP },
  },
  {
    id: 'luma-ray-2',
    modelId: 'ray-2',
    litellmId: 'luma/ray-2',
    name: 'ray-2',
    displayName: 'Ray 2',
    description: 'Latest model with improved realism',
    category: 'video_generation',
    capabilities: ['text_to_video', 'image_to_video', 'camera_motion'],
    inputModalities: ['text', 'image'],
    outputModalities: ['video'],
    pricing: { type: 'per_second', costPerSecond: 0.05, markup: MARKUP },
  },
];
```

```typescript
export const LUMA_PROVIDER: ExternalProvider = {
  id: 'luma',
  name: 'luma',
  displayName: 'Luma AI',
  category: 'video_generation',
  description: 'Cinematic AI video generation',
  website: 'https://lumalabs.ai',
  apiBaseUrl: 'https://api.lumalabs.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/luma',
  enabled: true,
  regions: ['us-east-1'],
  models: LUMA_MODELS,
  features: ['text_to_video', 'image_to_video', 'camera_control'],
};

// ============================================================================
// PIKA LABS
// ============================================================================

const PIKA_MODELS: ExternalModel[] = [
  {
    id: 'pika-v2',
    modelId: 'pika-2.0',
    litellmId: 'pika/pika-2.0',
    name: 'pika-2.0',
    displayName: 'Pika 2.0',
    description: 'Creative video generation with effects',
    category: 'video_generation',
    capabilities: ['text_to_video', 'image_to_video', 'video_editing'],
    inputModalities: ['text', 'image'],
    outputModalities: ['video'],
    pricing: { type: 'per_second', costPerSecond: 0.02, markup: MARKUP },
  },
];

export const PIKA_PROVIDER: ExternalProvider = {
  id: 'pika',
  name: 'pika',
  displayName: 'Pika Labs',
  category: 'video_generation',
  description: 'Creative AI video platform',
  website: 'https://pika.art',
  apiBaseUrl: 'https://api.pika.art/v1',
  authType: 'bearer',
```

```
  secretName: 'radiant/providers/pika',
  enabled: true,
  regions: ['us-east-1'],
  models: PIKA_MODELS,
  features: ['text_to_video', 'lip_sync', 'effects'],
};

// ============================================================================
// KLING AI
// ============================================================================

const KLING_MODELS: ExternalModel[] = [
  {
    id: 'kling-v15-pro',
    modelId: 'kling-v1.5-pro',
    litellmId: 'kling/kling-v1.5-pro',
    name: 'kling-v1.5-pro',
    displayName: 'Kling 1.5 Pro',
    description: 'High-quality Chinese video model',
    category: 'video_generation',
    capabilities: ['text_to_video', 'image_to_video'],
    inputModalities: ['text', 'image'],
    outputModalities: ['video'],
    pricing: { type: 'per_second', costPerSecond: 0.025, markup: MARKUP },
  },
];

export const KLING_PROVIDER: ExternalProvider = {
  id: 'kling',
  name: 'kling',
  displayName: 'Kling AI',
  category: 'video_generation',
  description: 'Chinese AI video generation',
  website: 'https://kling.kuaishou.com',
  apiBaseUrl: 'https://api.klingai.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/kling',
  enabled: true,
  regions: ['ap-southeast-1'],
  models: KLING_MODELS,
  features: ['text_to_video', 'image_to_video'],
};

// ============================================================================
// GOOGLE VEO
// ============================================================================
```

```typescript
const VEO_MODELS: ExternalModel[] = [
  {
    id: 'google-veo-2',
    modelId: 'veo-2.0',
    litellmId: 'gemini/veo-2.0',
    name: 'veo-2.0',
    displayName: 'Veo 2',
    description: 'Google DeepMind video generation',
    category: 'video_generation',
    capabilities: ['text_to_video', 'image_to_video'],
    inputModalities: ['text', 'image'],
    outputModalities: ['video'],
    pricing: { type: 'per_second', costPerSecond: 0.04, markup: MARKUP },
  },
];

export const VEO_PROVIDER: ExternalProvider = {
  id: 'veo',
  name: 'veo',
  displayName: 'Google Veo',
  category: 'video_generation',
  description: 'Google DeepMind video model',
  website: 'https://deepmind.google',
  apiBaseUrl: 'https://generativelanguage.googleapis.com/v1beta',
  authType: 'api_key',
  secretName: 'radiant/providers/google',
  enabled: true,
  regions: ['us-east-1'],
  models: VEO_MODELS,
  features: ['text_to_video', 'image_to_video'],
};

export const VIDEO_PROVIDERS: ExternalProvider[] = [
  RUNWAY_PROVIDER,
  LUMA_PROVIDER,
  PIKA_PROVIDER,
  KLING_PROVIDER,
  VEO_PROVIDER,
];
```

packages/infrastructure/lib/config/providers/audio.providers.ts

```typescript
/**
 * Audio/Speech Providers
 * OpenAI TTS/Whisper, ElevenLabs, Deepgram, AssemblyAI
```

```
 */

import { ExternalProvider, ExternalModel } from './index';

const MARKUP = 1.40;

// =============================================================================
// OPENAI AUDIO
// =============================================================================

const OPENAI_AUDIO_MODELS: ExternalModel[] = [
  {
    id: 'openai-tts-1',
    modelId: 'tts-1',
    litellmId: 'tts-1',
    name: 'tts-1',
    displayName: 'TTS-1',
    description: 'Fast text-to-speech',
    category: 'text_to_speech',
    capabilities: ['text_to_speech', 'multiple_voices'],
    inputModalities: ['text'],
    outputModalities: ['audio'],
    pricing: { type: 'per_token', inputCostPer1k: 0.015, markup: MARKUP, billedInputPer1k: (
  },
  {
    id: 'openai-tts-1-hd',
    modelId: 'tts-1-hd',
    litellmId: 'tts-1-hd',
    name: 'tts-1-hd',
    displayName: 'TTS-1 HD',
    description: 'High-definition text-to-speech',
    category: 'text_to_speech',
    capabilities: ['text_to_speech', 'multiple_voices', 'hd'],
    inputModalities: ['text'],
    outputModalities: ['audio'],
    pricing: { type: 'per_token', inputCostPer1k: 0.030, markup: MARKUP, billedInputPer1k: (
  },
  {
    id: 'openai-whisper-1',
    modelId: 'whisper-1',
    litellmId: 'whisper-1',
    name: 'whisper-1',
    displayName: 'Whisper',
    description: 'Speech recognition and transcription',
    category: 'speech_to_text',
    capabilities: ['transcription', 'translation', 'timestamps'],
```

```
    inputModalities: ['audio'],
    outputModalities: ['text'],
    pricing: { type: 'per_minute', costPerMinute: 0.006, markup: MARKUP },
  },
];

export const OPENAI_AUDIO_PROVIDER: ExternalProvider = {
  id: 'openai-audio',
  name: 'openai-audio',
  displayName: 'OpenAI Audio',
  category: 'audio_generation',
  description: 'OpenAI speech and audio models',
  website: 'https://openai.com',
  apiBaseUrl: 'https://api.openai.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/openai',
  enabled: true,
  regions: ['us-east-1'],
  models: OPENAI_AUDIO_MODELS,
  features: ['tts', 'stt', 'audio_understanding'],
};

// ============================================================================
// ELEVENLABS
// ============================================================================

const ELEVENLABS_MODELS: ExternalModel[] = [
  {
    id: 'elevenlabs-multilingual-v2',
    modelId: 'eleven_multilingual_v2',
    litellmId: 'elevenlabs/eleven_multilingual_v2',
    name: 'multilingual-v2',
    displayName: 'Multilingual V2',
    description: 'Multi-language voice synthesis',
    category: 'text_to_speech',
    capabilities: ['text_to_speech', 'multilingual', 'voice_cloning'],
    inputModalities: ['text'],
    outputModalities: ['audio'],
    pricing: { type: 'per_token', inputCostPer1k: 0.18, markup: MARKUP, billedInputPer1k: 0.
  },
  {
    id: 'elevenlabs-turbo-v2-5',
    modelId: 'eleven_turbo_v2_5',
    litellmId: 'elevenlabs/eleven_turbo_v2_5',
    name: 'turbo-v2.5',
    displayName: 'Turbo V2.5',
```

```
    description: 'Fast, low-latency voice synthesis',
    category: 'text_to_speech',
    capabilities: ['text_to_speech', 'low_latency', 'streaming'],
    inputModalities: ['text'],
    outputModalities: ['audio'],
    pricing: { type: 'per_token', inputCostPer1k: 0.09, markup: MARKUP, billedInputPer1k: 0.
  },
];

export const ELEVENLABS_PROVIDER: ExternalProvider = {
  id: 'elevenlabs',
  name: 'elevenlabs',
  displayName: 'ElevenLabs',
  category: 'text_to_speech',
  description: 'Premium voice synthesis and cloning',
  website: 'https://elevenlabs.io',
  apiBaseUrl: 'https://api.elevenlabs.io/v1',
  authType: 'api_key',
  secretName: 'radiant/providers/elevenlabs',
  enabled: true,
  regions: ['us-east-1'],
  models: ELEVENLABS_MODELS,
  features: ['voice_cloning', 'multilingual', 'streaming', 'sound_effects'],
};

// ============================================================================
// DEEPGRAM
// ============================================================================

const DEEPGRAM_MODELS: ExternalModel[] = [
  {
    id: 'deepgram-nova-2',
    modelId: 'nova-2',
    litellmId: 'deepgram/nova-2',
    name: 'nova-2',
    displayName: 'Nova-2',
    description: 'Industry-leading speech recognition',
    category: 'speech_to_text',
    capabilities: ['transcription', 'diarization', 'sentiment', 'streaming'],
    inputModalities: ['audio'],
    outputModalities: ['text'],
    pricing: { type: 'per_minute', costPerMinute: 0.0043, markup: MARKUP },
  },
  {
    id: 'deepgram-nova-2-medical',
    modelId: 'nova-2-medical',
```

```
    litellmId: 'deepgram/nova-2-medical',
    name: 'nova-2-medical',
    displayName: 'Nova-2 Medical',
    description: 'Medical terminology optimized',
    category: 'speech_to_text',
    capabilities: ['transcription', 'medical', 'hipaa'],
    inputModalities: ['audio'],
    outputModalities: ['text'],
    pricing: { type: 'per_minute', costPerMinute: 0.0079, markup: MARKUP },
  },
];

export const DEEPGRAM_PROVIDER: ExternalProvider = {
  id: 'deepgram',
  name: 'deepgram',
  displayName: 'Deepgram',
  category: 'speech_to_text',
  description: 'Enterprise speech recognition',
  website: 'https://deepgram.com',
  apiBaseUrl: 'https://api.deepgram.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/deepgram',
  enabled: true,
  regions: ['us-east-1'],
  models: DEEPGRAM_MODELS,
  features: ['real_time', 'diarization', 'sentiment', 'medical'],
  compliance: ['HIPAA', 'SOC2'],
};

// ============================================================================
// ASSEMBLYAI
// ============================================================================

const ASSEMBLYAI_MODELS: ExternalModel[] = [
  {
    id: 'assemblyai-best',
    modelId: 'best',
    litellmId: 'assemblyai/best',
    name: 'best',
    displayName: 'AssemblyAI Best',
    description: 'Highest accuracy transcription',
    category: 'speech_to_text',
    capabilities: ['transcription', 'diarization', 'summarization', 'chapters'],
    inputModalities: ['audio'],
    outputModalities: ['text'],
    pricing: { type: 'per_minute', costPerMinute: 0.0062, markup: MARKUP },
```

```
  },
];

export const ASSEMBLYAI_PROVIDER: ExternalProvider = {
  id: 'assemblyai',
  name: 'assemblyai',
  displayName: 'AssemblyAI',
  category: 'speech_to_text',
  description: 'AI-powered transcription and audio intelligence',
  website: 'https://assemblyai.com',
  apiBaseUrl: 'https://api.assemblyai.com/v2',
  authType: 'bearer',
  secretName: 'radiant/providers/assemblyai',
  enabled: true,
  regions: ['us-east-1'],
  models: ASSEMBLYAI_MODELS,
  features: ['lemur', 'summarization', 'chapters', 'entity_detection'],
  compliance: ['SOC2', 'GDPR'],
};

export const AUDIO_PROVIDERS: ExternalProvider[] = [
  OPENAI_AUDIO_PROVIDER,
  ELEVENLABS_PROVIDER,
  DEEPGRAM_PROVIDER,
  ASSEMBLYAI_PROVIDER,
];
```

packages/infrastructure/lib/config/providers/embedding.providers.ts

```
/**
 * Embedding Providers
 * OpenAI, Cohere, Voyage, Google
 */

import { ExternalProvider, ExternalModel } from './index';

const MARKUP = 1.40;

const OPENAI_EMBEDDING_MODELS: ExternalModel[] = [
  {
    id: 'openai-text-embedding-3-small',
    modelId: 'text-embedding-3-small',
    litellmId: 'text-embedding-3-small',
    name: 'text-embedding-3-small',
    displayName: 'Text Embedding 3 Small',
    description: 'Efficient embedding with 1536 dimensions',
```

```
    category: 'embedding',
    capabilities: ['text_embedding', 'variable_dimensions'],
    contextWindow: 8191,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_token', inputCostPer1k: 0.00002, markup: MARKUP, billedInputPer1k:
  },
  {
    id: 'openai-text-embedding-3-large',
    modelId: 'text-embedding-3-large',
    litellmId: 'text-embedding-3-large',
    name: 'text-embedding-3-large',
    displayName: 'Text Embedding 3 Large',
    description: 'High-performance with 3072 dimensions',
    category: 'embedding',
    capabilities: ['text_embedding', 'variable_dimensions'],
    contextWindow: 8191,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_token', inputCostPer1k: 0.00013, markup: MARKUP, billedInputPer1k:
  },
];

export const OPENAI_EMBEDDING_PROVIDER: ExternalProvider = {
  id: 'openai-embeddings',
  name: 'openai-embeddings',
  displayName: 'OpenAI Embeddings',
  category: 'embedding',
  description: 'OpenAI text embedding models',
  website: 'https://openai.com',
  apiBaseUrl: 'https://api.openai.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/openai',
  enabled: true,
  regions: ['us-east-1'],
  models: OPENAI_EMBEDDING_MODELS,
  features: ['variable_dimensions', 'batch'],
};

const VOYAGE_MODELS: ExternalModel[] = [
  {
    id: 'voyage-3',
    modelId: 'voyage-3',
    litellmId: 'voyage/voyage-3',
    name: 'voyage-3',
    displayName: 'Voyage 3',
```

```
    description: 'State-of-the-art general-purpose embeddings',
    category: 'embedding',
    capabilities: ['text_embedding'],
    contextWindow: 32000,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_token', inputCostPer1k: 0.00006, markup: MARKUP, billedInputPer1k:
  },
  {
    id: 'voyage-code-3',
    modelId: 'voyage-code-3',
    litellmId: 'voyage/voyage-code-3',
    name: 'voyage-code-3',
    displayName: 'Voyage Code 3',
    description: 'Code-optimized embeddings',
    category: 'embedding',
    capabilities: ['text_embedding', 'code'],
    contextWindow: 32000,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_token', inputCostPer1k: 0.00006, markup: MARKUP, billedInputPer1k:
  },
];

export const VOYAGE_PROVIDER: ExternalProvider = {
  id: 'voyage',
  name: 'voyage',
  displayName: 'Voyage AI',
  category: 'embedding',
  description: 'Premium embedding models',
  website: 'https://voyageai.com',
  apiBaseUrl: 'https://api.voyageai.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/voyage',
  enabled: true,
  regions: ['us-east-1'],
  models: VOYAGE_MODELS,
  features: ['long_context', 'code', 'multimodal'],
};

export const EMBEDDING_PROVIDERS: ExternalProvider[] = [
  OPENAI_EMBEDDING_PROVIDER,
  VOYAGE_PROVIDER,
];
```

packages/infrastructure/lib/config/providers/search.providers.ts

```
/**
 * Search Providers
 * Perplexity, Exa, Tavily
 */

import { ExternalProvider, ExternalModel } from './index';

const MARKUP = 1.40;

const PERPLEXITY_MODELS: ExternalModel[] = [
  {
    id: 'perplexity-sonar-pro',
    modelId: 'sonar-pro',
    litellmId: 'perplexity/sonar-pro',
    name: 'sonar-pro',
    displayName: 'Sonar Pro',
    description: 'Premium search-augmented model',
    category: 'search',
    capabilities: ['search', 'citations', 'reasoning'],
    contextWindow: 200000,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_token', inputCostPer1k: 0.003, outputCostPer1k: 0.015, baseCostPer
  },
  {
    id: 'perplexity-sonar',
    modelId: 'sonar',
    litellmId: 'perplexity/sonar',
    name: 'sonar',
    displayName: 'Sonar',
    description: 'Fast search-augmented model',
    category: 'search',
    capabilities: ['search', 'citations'],
    contextWindow: 128000,
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_token', inputCostPer1k: 0.001, outputCostPer1k: 0.001, baseCostPer
  },
];

export const PERPLEXITY_PROVIDER: ExternalProvider = {
  id: 'perplexity',
  name: 'perplexity',
  displayName: 'Perplexity',
```

```
  category: 'search',
  description: 'AI search engine with real-time web access',
  website: 'https://perplexity.ai',
  apiBaseUrl: 'https://api.perplexity.ai',
  authType: 'bearer',
  secretName: 'radiant/providers/perplexity',
  enabled: true,
  regions: ['us-east-1'],
  models: PERPLEXITY_MODELS,
  features: ['web_search', 'citations', 'deep_research'],
};

const EXA_MODELS: ExternalModel[] = [
  {
    id: 'exa-search',
    modelId: 'exa-search',
    litellmId: 'exa/search',
    name: 'exa-search',
    displayName: 'Exa Search',
    description: 'Neural search engine for precise results',
    category: 'search',
    capabilities: ['neural_search', 'content_extraction'],
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_request', baseCostPerRequest: 0.001, markup: MARKUP },
  },
];

export const EXA_PROVIDER: ExternalProvider = {
  id: 'exa',
  name: 'exa',
  displayName: 'Exa',
  category: 'search',
  description: 'Neural search API',
  website: 'https://exa.ai',
  apiBaseUrl: 'https://api.exa.ai/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/exa',
  enabled: true,
  regions: ['us-east-1'],
  models: EXA_MODELS,
  features: ['neural_search', 'similarity', 'keyword'],
};

const TAVILY_MODELS: ExternalModel[] = [
  {
```

```
    id: 'tavily-search',
    modelId: 'tavily-search',
    litellmId: 'tavily/search',
    name: 'tavily-search',
    displayName: 'Tavily Search',
    description: 'AI-optimized search API',
    category: 'search',
    capabilities: ['search', 'answer_generation'],
    inputModalities: ['text'],
    outputModalities: ['text'],
    pricing: { type: 'per_request', baseCostPerRequest: 0.001, markup: MARKUP },
  },
];

export const TAVILY_PROVIDER: ExternalProvider = {
  id: 'tavily',
  name: 'tavily',
  displayName: 'Tavily',
  category: 'search',
  description: 'Search API for AI agents',
  website: 'https://tavily.com',
  apiBaseUrl: 'https://api.tavily.com/v1',
  authType: 'bearer',
  secretName: 'radiant/providers/tavily',
  enabled: true,
  regions: ['us-east-1'],
  models: TAVILY_MODELS,
  features: ['search', 'extract', 'answer'],
};

export const SEARCH_PROVIDERS: ExternalProvider[] = [
  PERPLEXITY_PROVIDER,
  EXA_PROVIDER,
  TAVILY_PROVIDER,
];
```

packages/infrastructure/lib/config/providers/3d.providers.ts

```
/**
 * 3D Generation Providers
 * Meshy, Rodin, Tripo
 */

import { ExternalProvider, ExternalModel } from './index';

const MARKUP = 1.40;
```

```typescript
const MESHY_MODELS: ExternalModel[] = [
  {
    id: 'meshy-v3',
    modelId: 'meshy-v3',
    litellmId: 'meshy/meshy-v3',
    name: 'meshy-v3',
    displayName: 'Meshy V3',
    description: 'Text-to-3D and image-to-3D generation',
    category: '3d_generation',
    capabilities: ['text_to_3d', 'image_to_3d'],
    inputModalities: ['text', 'image'],
    outputModalities: ['3d'],
    pricing: { type: 'per_request', baseCostPerRequest: 0.20, markup: MARKUP },
  },
  {
    id: 'meshy-v3-turbo',
    modelId: 'meshy-v3-turbo',
    litellmId: 'meshy/meshy-v3-turbo',
    name: 'meshy-v3-turbo',
    displayName: 'Meshy V3 Turbo',
    description: 'Fast 3D generation (preview quality)',
    category: '3d_generation',
    capabilities: ['text_to_3d', 'fast'],
    inputModalities: ['text'],
    outputModalities: ['3d'],
    pricing: { type: 'per_request', baseCostPerRequest: 0.05, markup: MARKUP },
  },
];

export const MESHY_PROVIDER: ExternalProvider = {
  id: 'meshy',
  name: 'meshy',
  displayName: 'Meshy',
  category: '3d_generation',
  description: 'AI 3D model generation',
  website: 'https://meshy.ai',
  apiBaseUrl: 'https://api.meshy.ai/v2',
  authType: 'bearer',
  secretName: 'radiant/providers/meshy',
  enabled: true,
  regions: ['us-east-1'],
  models: MESHY_MODELS,
  features: ['text_to_3d', 'image_to_3d', 'texturing'],
};
```

```typescript
const TRIPO_MODELS: ExternalModel[] = [
  {
    id: 'tripo-v2',
    modelId: 'tripo-v2',
    litellmId: 'tripo/tripo-v2',
    name: 'tripo-v2',
    displayName: 'Tripo V2',
    description: 'Fast text and image to 3D',
    category: '3d_generation',
    capabilities: ['text_to_3d', 'image_to_3d'],
    inputModalities: ['text', 'image'],
    outputModalities: ['3d'],
    pricing: { type: 'per_request', baseCostPerRequest: 0.10, markup: MARKUP },
  },
];

export const TRIPO_PROVIDER: ExternalProvider = {
  id: 'tripo',
  name: 'tripo',
  displayName: 'Tripo AI',
  category: '3d_generation',
  description: 'AI 3D generation platform',
  website: 'https://tripo3d.ai',
  apiBaseUrl: 'https://api.tripo3d.ai/v2',
  authType: 'bearer',
  secretName: 'radiant/providers/tripo',
  enabled: true,
  regions: ['us-east-1'],
  models: TRIPO_MODELS,
  features: ['text_to_3d', 'image_to_3d', 'animation'],
};

export const THREE_D_PROVIDERS: ExternalProvider[] = [
  MESHY_PROVIDER,
  TRIPO_PROVIDER,
];
```

PART 2: LITELLM CONFIGURATION

packages/infrastructure/litellm/config/external.yaml

```yaml
# LiteLLM Configuration - External Providers
# RADIANT v2.2.0 - December 2024

model_list:
```

```yaml
# ==============================================================================
# OPENAI MODELS
# ==============================================================================
- model_name: gpt-4o
  litellm_params:
    model: gpt-4o
    api_key: os.environ/OPENAI_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.0000025
    output_cost_per_token: 0.00001
    max_tokens: 16384
    max_input_tokens: 128000
    supports_function_calling: true
    supports_vision: true

- model_name: gpt-4o-mini
  litellm_params:
    model: gpt-4o-mini
    api_key: os.environ/OPENAI_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.00000015
    output_cost_per_token: 0.0000006
    max_tokens: 16384
    max_input_tokens: 128000

- model_name: o1
  litellm_params:
    model: o1
    api_key: os.environ/OPENAI_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.000015
    output_cost_per_token: 0.00006
    max_tokens: 100000
    max_input_tokens: 200000

- model_name: o3-mini
  litellm_params:
    model: o3-mini
    api_key: os.environ/OPENAI_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.0000011
    output_cost_per_token: 0.0000044
```

```yaml
# ==============================================================================
# ANTHROPIC MODELS
# ==============================================================================
- model_name: claude-opus-4
  litellm_params:
    model: anthropic/claude-opus-4-20250514
    api_key: os.environ/ANTHROPIC_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.000015
    output_cost_per_token: 0.000075
    max_tokens: 32000
    max_input_tokens: 200000

- model_name: claude-sonnet-4
  litellm_params:
    model: anthropic/claude-sonnet-4-20250514
    api_key: os.environ/ANTHROPIC_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.000003
    output_cost_per_token: 0.000015
    max_tokens: 64000
    max_input_tokens: 200000

- model_name: claude-3.5-haiku
  litellm_params:
    model: anthropic/claude-3-5-haiku-20241022
    api_key: os.environ/ANTHROPIC_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.0000008
    output_cost_per_token: 0.000004

# ==============================================================================
# GOOGLE MODELS
# ==============================================================================
- model_name: gemini-2.0-flash
  litellm_params:
    model: gemini/gemini-2.0-flash
    api_key: os.environ/GOOGLE_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.0000001
    output_cost_per_token: 0.0000004
```

```yaml
    max_input_tokens: 1000000

- model_name: gemini-1.5-pro
  litellm_params:
    model: gemini/gemini-1.5-pro
    api_key: os.environ/GOOGLE_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.00000125
    output_cost_per_token: 0.000005
    max_input_tokens: 2000000

# ============================================================================
# XAI (GROK) MODELS
# ============================================================================
- model_name: grok-3
  litellm_params:
    model: xai/grok-3
    api_key: os.environ/XAI_API_KEY
    api_base: https://api.x.ai/v1
  model_info:
    mode: chat
    input_cost_per_token: 0.000003
    output_cost_per_token: 0.000015

# ============================================================================
# DEEPSEEK MODELS
# ============================================================================
- model_name: deepseek-chat
  litellm_params:
    model: deepseek/deepseek-chat
    api_key: os.environ/DEEPSEEK_API_KEY
    api_base: https://api.deepseek.com/v1
  model_info:
    mode: chat
    input_cost_per_token: 0.00000027
    output_cost_per_token: 0.0000011

- model_name: deepseek-reasoner
  litellm_params:
    model: deepseek/deepseek-reasoner
    api_key: os.environ/DEEPSEEK_API_KEY
    api_base: https://api.deepseek.com/v1
  model_info:
    mode: chat
    input_cost_per_token: 0.00000055
```

```yaml
    output_cost_per_token: 0.00000219

# ============================================================================
# MISTRAL MODELS
# ============================================================================
- model_name: mistral-large
  litellm_params:
    model: mistral/mistral-large-latest
    api_key: os.environ/MISTRAL_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.000002
    output_cost_per_token: 0.000006

- model_name: mistral-small
  litellm_params:
    model: mistral/mistral-small-latest
    api_key: os.environ/MISTRAL_API_KEY
  model_info:
    mode: chat
    input_cost_per_token: 0.0000002
    output_cost_per_token: 0.0000006

# ============================================================================
# EMBEDDING MODELS
# ============================================================================
- model_name: text-embedding-3-small
  litellm_params:
    model: text-embedding-3-small
    api_key: os.environ/OPENAI_API_KEY
  model_info:
    mode: embedding
    input_cost_per_token: 0.00000002

- model_name: text-embedding-3-large
  litellm_params:
    model: text-embedding-3-large
    api_key: os.environ/OPENAI_API_KEY
  model_info:
    mode: embedding
    input_cost_per_token: 0.00000013

- model_name: voyage-3
  litellm_params:
    model: voyage/voyage-3
    api_key: os.environ/VOYAGE_API_KEY
```

```yaml
    model_info:
      mode: embedding
      input_cost_per_token: 0.00000006

  # ============================================================================
  # PERPLEXITY SEARCH MODELS
  # ============================================================================
  - model_name: sonar-pro
    litellm_params:
      model: perplexity/sonar-pro
      api_key: os.environ/PERPLEXITY_API_KEY
    model_info:
      mode: chat
      input_cost_per_token: 0.000003
      output_cost_per_token: 0.000015

  - model_name: sonar
    litellm_params:
      model: perplexity/sonar
      api_key: os.environ/PERPLEXITY_API_KEY
    model_info:
      mode: chat
      input_cost_per_token: 0.000001
      output_cost_per_token: 0.000001

# ============================================================================
# LITELLM SETTINGS
# ============================================================================
litellm_settings:
  success_callback: ["langfuse"]
  failure_callback: ["langfuse"]
  request_timeout: 300
  num_retries: 3
  retry_after: 5
  cache: true
  cache_params:
    type: redis
    host: os.environ/REDIS_HOST
    port: 6379
    ttl: 3600
  enable_rate_limiting: true
  set_verbose: false
  json_logs: true

# ============================================================================
# ROUTER SETTINGS
```

```yaml
# ============================================================================
router_settings:
  routing_strategy: "least-busy"
  enable_pre_call_checks: true
  allowed_fails: 3
  cooldown_time: 60
  fallbacks:
    - model_name: gpt-4o
      fallback_models: ["claude-sonnet-4", "gemini-2.0-flash"]
    - model_name: claude-opus-4
      fallback_models: ["gpt-4o", "gemini-1.5-pro"]

general_settings:
  master_key: os.environ/LITELLM_MASTER_KEY
  database_url: os.environ/LITELLM_DATABASE_URL
  alerting: ["slack"]
  alerting_threshold: 300
```

PART 3: POSTGRESQL SCHEMA

packages/infrastructure/migrations/001_initial_schema.sql

```sql
-- Migration: 001_initial_schema
-- RADIANT v2.2.0 - Base tables

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "pgcrypto";
CREATE EXTENSION IF NOT EXISTS "pg_trgm";

-- ============================================================================
-- TENANTS
-- ============================================================================

CREATE TABLE tenants (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    app_id VARCHAR(50) NOT NULL,
    name VARCHAR(255) NOT NULL,
    slug VARCHAR(100) NOT NULL UNIQUE,
    domain VARCHAR(255),
    tier INTEGER NOT NULL DEFAULT 1 CHECK (tier BETWEEN 1 AND 5),
    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'suspended', '
    stripe_customer_id VARCHAR(100),
    billing_email VARCHAR(255),
    settings JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
```

```
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    deleted_at TIMESTAMPTZ
);

CREATE INDEX idx_tenants_app_id ON tenants(app_id);
CREATE INDEX idx_tenants_slug ON tenants(slug);
CREATE INDEX idx_tenants_status ON tenants(status);

-- ============================================================================
-- USERS
-- ============================================================================

CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    cognito_sub VARCHAR(100) UNIQUE,
    email VARCHAR(255) NOT NULL,
    email_verified BOOLEAN DEFAULT false,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    display_name VARCHAR(200),
    avatar_url TEXT,
    preferences JSONB DEFAULT '{}',
    timezone VARCHAR(50) DEFAULT 'UTC',
    locale VARCHAR(10) DEFAULT 'en-US',
    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'su
    last_login_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    deleted_at TIMESTAMPTZ,
    UNIQUE(tenant_id, email)
);

CREATE INDEX idx_users_tenant ON users(tenant_id);
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_cognito ON users(cognito_sub);

-- ============================================================================
-- API KEYS
-- ============================================================================

CREATE TABLE api_keys (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    name VARCHAR(100) NOT NULL,
```

```
    key_prefix VARCHAR(10) NOT NULL,
    key_hash VARCHAR(255) NOT NULL,
    scopes TEXT[] DEFAULT ARRAY['chat', 'models', 'embeddings'],
    rate_limit_rpm INTEGER DEFAULT 60,
    rate_limit_tpm INTEGER DEFAULT 100000,
    last_used_at TIMESTAMPTZ,
    usage_count BIGINT DEFAULT 0,
    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'revoked', 'exp
    expires_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_api_keys_tenant ON api_keys(tenant_id);
CREATE INDEX idx_api_keys_prefix ON api_keys(key_prefix);
CREATE UNIQUE INDEX idx_api_keys_hash ON api_keys(key_hash);

-- ============================================================================
-- SCHEMA MIGRATIONS TRACKING
-- ============================================================================

CREATE TABLE schema_migrations (
    version VARCHAR(20) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    applied_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    applied_by VARCHAR(100),
    checksum VARCHAR(64)
);

INSERT INTO schema_migrations (version, name, applied_by)
VALUES ('001', 'initial_schema', 'system');
```

packages/infrastructure/migrations/002_tenant_isolation.sql

```
-- Migration: 002_tenant_isolation
-- Row-Level Security policies

ALTER TABLE tenants ENABLE ROW LEVEL SECURITY;
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
ALTER TABLE api_keys ENABLE ROW LEVEL SECURITY;

CREATE OR REPLACE FUNCTION current_tenant_id() RETURNS UUID AS $$
BEGIN
    RETURN NULLIF(current_setting('app.current_tenant_id', true), '')::UUID;
EXCEPTION WHEN OTHERS THEN RETURN NULL;
END;
```

```
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE POLICY tenant_isolation_select ON tenants FOR SELECT USING (id = current_tenant_id())
CREATE POLICY tenant_isolation_update ON tenants FOR UPDATE USING (id = current_tenant_id())

CREATE POLICY users_tenant_isolation ON users FOR ALL USING (tenant_id = current_tenant_id()
CREATE POLICY api_keys_tenant_isolation ON api_keys FOR ALL USING (tenant_id = current_tenan

-- System role for admin operations
DO $$ BEGIN IF NOT EXISTS (SELECT FROM pg_roles WHERE rolname = 'radiant_system') THEN CREAT
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO radiant_system;

INSERT INTO schema_migrations (version, name, applied_by) VALUES ('002', 'tenant_isolation',
```

packages/infrastructure/migrations/003_ai_models.sql

```
-- Migration: 003_ai_models
-- Providers and models registry

CREATE TABLE providers (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    display_name VARCHAR(100) NOT NULL,
    category VARCHAR(50) NOT NULL,
    description TEXT,
    website VARCHAR(255),
    api_base_url VARCHAR(255),
    auth_type VARCHAR(20) NOT NULL DEFAULT 'bearer',
    secret_name VARCHAR(100),
    enabled BOOLEAN DEFAULT true,
    features TEXT[],
    regions TEXT[],
    compliance TEXT[],
    rate_limit_rpm INTEGER,
    rate_limit_tpm INTEGER,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_providers_category ON providers(category);
CREATE INDEX idx_providers_enabled ON providers(enabled);

CREATE TABLE models (
    id VARCHAR(100) PRIMARY KEY,
    provider_id VARCHAR(50) NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
```

```
    model_id VARCHAR(100) NOT NULL,
    litellm_id VARCHAR(100) NOT NULL UNIQUE,
    name VARCHAR(100) NOT NULL,
    display_name VARCHAR(100) NOT NULL,
    description TEXT,
    category VARCHAR(50) NOT NULL,
    capabilities TEXT[],
    context_window INTEGER,
    max_output INTEGER,
    input_modalities TEXT[],
    output_modalities TEXT[],
    input_cost_per_1k NUMERIC(10, 8),
    output_cost_per_1k NUMERIC(10, 8),
    cost_per_request NUMERIC(10, 6),
    cost_per_second NUMERIC(10, 6),
    cost_per_image NUMERIC(10, 6),
    cost_per_minute NUMERIC(10, 6),
    pricing_type VARCHAR(20) NOT NULL DEFAULT 'per_token',
    markup_rate NUMERIC(4, 2) NOT NULL DEFAULT 1.40,
    enabled BOOLEAN DEFAULT true,
    deprecated BOOLEAN DEFAULT false,
    is_self_hosted BOOLEAN DEFAULT false,
    self_hosted_config JSONB,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_models_provider ON models(provider_id);
CREATE INDEX idx_models_category ON models(category);
CREATE INDEX idx_models_enabled ON models(enabled);
CREATE INDEX idx_models_litellm ON models(litellm_id);

CREATE TABLE tenant_model_access (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    model_id VARCHAR(100) NOT NULL REFERENCES models(id) ON DELETE CASCADE,
    enabled BOOLEAN DEFAULT true,
    custom_input_cost_per_1k NUMERIC(10, 8),
    custom_output_cost_per_1k NUMERIC(10, 8),
    custom_markup_rate NUMERIC(4, 2),
    rate_limit_rpm INTEGER,
    rate_limit_tpm INTEGER,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    UNIQUE(tenant_id, model_id)
```

```
);

ALTER TABLE tenant_model_access ENABLE ROW LEVEL SECURITY;
CREATE POLICY tenant_model_access_isolation ON tenant_model_access FOR ALL USING (tenant_id

INSERT INTO schema_migrations (version, name, applied_by) VALUES ('003', 'ai_models', 'syste
```

packages/infrastructure/migrations/004_usage_billing.sql

```
-- Migration: 004_usage_billing
-- Usage tracking and billing

CREATE TABLE usage_events (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    api_key_id UUID REFERENCES api_keys(id) ON DELETE SET NULL,
    request_id VARCHAR(100) NOT NULL,
    model_id VARCHAR(100) NOT NULL REFERENCES models(id),
    provider_id VARCHAR(50) NOT NULL REFERENCES providers(id),
    input_tokens INTEGER NOT NULL DEFAULT 0,
    output_tokens INTEGER NOT NULL DEFAULT 0,
    total_tokens INTEGER GENERATED ALWAYS AS (input_tokens + output_tokens) STORED,
    image_count INTEGER DEFAULT 0,
    video_seconds NUMERIC(10, 2) DEFAULT 0,
    audio_minutes NUMERIC(10, 2) DEFAULT 0,
    request_count INTEGER DEFAULT 1,
    base_cost NUMERIC(12, 8) NOT NULL DEFAULT 0,
    billed_cost NUMERIC(12, 8) NOT NULL DEFAULT 0,
    endpoint VARCHAR(50),
    latency_ms INTEGER,
    status VARCHAR(20) NOT NULL DEFAULT 'success',
    error_code VARCHAR(50),
    ip_address INET,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_usage_events_tenant ON usage_events(tenant_id);
CREATE INDEX idx_usage_events_model ON usage_events(model_id);
CREATE INDEX idx_usage_events_created ON usage_events(created_at DESC);
CREATE INDEX idx_usage_events_request ON usage_events(request_id);

ALTER TABLE usage_events ENABLE ROW LEVEL SECURITY;
CREATE POLICY usage_events_tenant_isolation ON usage_events FOR ALL USING (tenant_id = curre
```

```sql
CREATE TABLE usage_rollups (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    rollup_date DATE NOT NULL,
    model_id VARCHAR(100) REFERENCES models(id),
    provider_id VARCHAR(50) REFERENCES providers(id),
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    request_count BIGINT NOT NULL DEFAULT 0,
    success_count BIGINT NOT NULL DEFAULT 0,
    error_count BIGINT NOT NULL DEFAULT 0,
    input_tokens BIGINT NOT NULL DEFAULT 0,
    output_tokens BIGINT NOT NULL DEFAULT 0,
    total_tokens BIGINT NOT NULL DEFAULT 0,
    image_count INTEGER DEFAULT 0,
    video_seconds NUMERIC(12, 2) DEFAULT 0,
    audio_minutes NUMERIC(12, 2) DEFAULT 0,
    base_cost NUMERIC(14, 6) NOT NULL DEFAULT 0,
    billed_cost NUMERIC(14, 6) NOT NULL DEFAULT 0,
    avg_latency_ms NUMERIC(10, 2),
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    UNIQUE(tenant_id, rollup_date, model_id, provider_id, user_id)
);

CREATE INDEX idx_usage_rollups_tenant ON usage_rollups(tenant_id);
CREATE INDEX idx_usage_rollups_date ON usage_rollups(rollup_date DESC);

ALTER TABLE usage_rollups ENABLE ROW LEVEL SECURITY;
CREATE POLICY usage_rollups_tenant_isolation ON usage_rollups FOR ALL USING (tenant_id = cur

CREATE TABLE invoices (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    invoice_number VARCHAR(50) NOT NULL UNIQUE,
    billing_period_start DATE NOT NULL,
    billing_period_end DATE NOT NULL,
    subtotal NUMERIC(14, 2) NOT NULL DEFAULT 0,
    discount_amount NUMERIC(14, 2) NOT NULL DEFAULT 0,
    tax_amount NUMERIC(14, 2) NOT NULL DEFAULT 0,
    total_amount NUMERIC(14, 2) NOT NULL DEFAULT 0,
    status VARCHAR(20) NOT NULL DEFAULT 'draft' CHECK (status IN ('draft', 'pending', 'paid'
    stripe_invoice_id VARCHAR(100),
    paid_at TIMESTAMPTZ,
    line_items JSONB NOT NULL DEFAULT '[]',
    notes TEXT,
    metadata JSONB DEFAULT '{}',
```

```
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_invoices_tenant ON invoices(tenant_id);
CREATE INDEX idx_invoices_status ON invoices(status);

ALTER TABLE invoices ENABLE ROW LEVEL SECURITY;
CREATE POLICY invoices_tenant_isolation ON invoices FOR ALL USING (tenant_id = current_tenan

INSERT INTO schema_migrations (version, name, applied_by) VALUES ('004', 'usage_billing', 's
```

packages/infrastructure/migrations/005_admin_approval.sql

```
-- Migration: 005_admin_approval
-- Administrator management and approval workflows

CREATE TABLE administrators (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    cognito_sub VARCHAR(100) UNIQUE,
    email VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    display_name VARCHAR(200),
    phone VARCHAR(20),
    role VARCHAR(20) NOT NULL CHECK (role IN ('super_admin', 'admin', 'operator', 'auditor')
    custom_permissions TEXT[],
    mfa_enabled BOOLEAN DEFAULT false,
    mfa_method VARCHAR(20) CHECK (mfa_method IN ('authenticator', 'sms', 'email')),
    mfa_verified_at TIMESTAMPTZ,
    preferences JSONB DEFAULT '{}',
    notification_settings JSONB DEFAULT '{"email": true, "slack": false, "approval_required"
    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('pending', 'active', 'ina
    last_login_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    deleted_at TIMESTAMPTZ,
    UNIQUE(tenant_id, email)
);

CREATE INDEX idx_admins_tenant ON administrators(tenant_id);
CREATE INDEX idx_admins_email ON administrators(email);
CREATE INDEX idx_admins_role ON administrators(role);

ALTER TABLE administrators ENABLE ROW LEVEL SECURITY;
```

```
CREATE POLICY admins_tenant_isolation ON administrators FOR ALL USING (tenant_id = current_t

CREATE TABLE invitations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    email VARCHAR(255) NOT NULL,
    role VARCHAR(20) NOT NULL CHECK (role IN ('super_admin', 'admin', 'operator', 'auditor')
    message TEXT,
    token_hash VARCHAR(255) NOT NULL,
    invited_by UUID NOT NULL REFERENCES administrators(id),
    invited_by_name VARCHAR(200) NOT NULL,
    status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (status IN ('pending', 'accepted', '
    expires_at TIMESTAMPTZ NOT NULL,
    accepted_at TIMESTAMPTZ,
    accepted_by_ip INET,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_invitations_tenant ON invitations(tenant_id);
CREATE INDEX idx_invitations_email ON invitations(email);
CREATE INDEX idx_invitations_status ON invitations(status);
CREATE INDEX idx_invitations_token ON invitations(token_hash);

ALTER TABLE invitations ENABLE ROW LEVEL SECURITY;
CREATE POLICY invitations_tenant_isolation ON invitations FOR ALL USING (tenant_id = current

CREATE TABLE approval_requests (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    app_id VARCHAR(50) NOT NULL,
    environment VARCHAR(20) NOT NULL CHECK (environment IN ('dev', 'staging', 'prod')),
    type VARCHAR(50) NOT NULL CHECK (type IN ('deployment', 'promotion', 'model_activation',
    title VARCHAR(255) NOT NULL,
    description TEXT,
    payload JSONB NOT NULL DEFAULT '{}',
    risk_level VARCHAR(20) NOT NULL DEFAULT 'medium' CHECK (risk_level IN ('low', 'medium',
    impact_analysis TEXT,
    rollback_plan TEXT,
    initiated_by UUID NOT NULL REFERENCES administrators(id),
    initiated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (status IN ('pending', 'approved', '
    approved_by UUID REFERENCES administrators(id),
    approved_at TIMESTAMPTZ,
    approval_notes TEXT,
    executed_at TIMESTAMPTZ,
    execution_result JSONB,
```

```
    expires_at TIMESTAMPTZ NOT NULL DEFAULT (NOW() + INTERVAL '24 hours'),
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    CONSTRAINT different_approver CHECK (approved_by IS NULL OR approved_by != initiated_by)
);

CREATE INDEX idx_approval_tenant ON approval_requests(tenant_id);
CREATE INDEX idx_approval_status ON approval_requests(status);
CREATE INDEX idx_approval_environment ON approval_requests(environment);

ALTER TABLE approval_requests ENABLE ROW LEVEL SECURITY;
CREATE POLICY approval_tenant_isolation ON approval_requests FOR ALL USING (tenant_id = curr

CREATE TABLE audit_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    admin_id UUID REFERENCES administrators(id) ON DELETE SET NULL,
    action VARCHAR(100) NOT NULL,
    resource_type VARCHAR(100) NOT NULL,
    resource_id VARCHAR(255),
    ip_address INET,
    user_agent TEXT,
    request_id VARCHAR(100),
    old_values JSONB,
    new_values JSONB,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_audit_tenant ON audit_logs(tenant_id);
CREATE INDEX idx_audit_action ON audit_logs(action);
CREATE INDEX idx_audit_resource ON audit_logs(resource_type, resource_id);
CREATE INDEX idx_audit_created ON audit_logs(created_at DESC);

ALTER TABLE audit_logs ENABLE ROW LEVEL SECURITY;
CREATE POLICY audit_logs_tenant_select ON audit_logs FOR SELECT USING (tenant_id = current_t

INSERT INTO schema_migrations (version, name, applied_by) VALUES ('005', 'admin_approval', '
```

packages/infrastructure/migrations/007_external_providers.sql

```
-- Migration: 007_external_providers
-- External provider configuration

CREATE TABLE provider_health (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    provider_id VARCHAR(50) NOT NULL REFERENCES providers(id) ON DELETE CASCADE,
    region VARCHAR(20) NOT NULL,
    status VARCHAR(20) NOT NULL DEFAULT 'healthy' CHECK (status IN ('healthy', 'degraded', '
    latency_ms INTEGER,
    error_rate NUMERIC(5, 2),
    success_rate NUMERIC(5, 2),
    last_check_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    last_success_at TIMESTAMPTZ,
    last_failure_at TIMESTAMPTZ,
    last_error TEXT,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    UNIQUE(provider_id, region)
);

CREATE INDEX idx_provider_health_provider ON provider_health(provider_id);
CREATE INDEX idx_provider_health_status ON provider_health(status);

CREATE TABLE routing_rules (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id) ON DELETE CASCADE,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    priority INTEGER NOT NULL DEFAULT 100,
    enabled BOOLEAN DEFAULT true,
    conditions JSONB NOT NULL DEFAULT '{}',
    action VARCHAR(50) NOT NULL CHECK (action IN ('route', 'fallback', 'block', 'rate_limit'
    target_provider VARCHAR(50),
    target_model VARCHAR(100),
    fallback_models TEXT[],
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_routing_rules_tenant ON routing_rules(tenant_id);
CREATE INDEX idx_routing_rules_priority ON routing_rules(priority);
CREATE INDEX idx_routing_rules_enabled ON routing_rules(enabled);

-- Populate initial providers
INSERT INTO providers (id, name, display_name, category, description, website, api_base_url,
('openai', 'openai', 'OpenAI', 'text_generation', 'GPT and O-series models', 'https://openai
('anthropic', 'anthropic', 'Anthropic', 'text_generation', 'Claude models', 'https://anthrop
('google', 'google', 'Google AI', 'text_generation', 'Gemini models', 'https://ai.google.dev
('xai', 'xai', 'xAI', 'text_generation', 'Grok models', 'https://x.ai', 'https://api.x.ai/v1
('deepseek', 'deepseek', 'DeepSeek', 'text_generation', 'Affordable models', 'https://deepse
```

```
('mistral', 'mistral', 'Mistral AI', 'text_generation', 'European models', 'https://mistral.
('cohere', 'cohere', 'Cohere', 'text_generation', 'Enterprise AI', 'https://cohere.com', 'ht
('stability', 'stability', 'Stability AI', 'image_generation', 'Open-source images', 'https:
('runway', 'runway', 'Runway', 'video_generation', 'AI video tools', 'https://runway.ml', 'h
('luma', 'luma', 'Luma AI', 'video_generation', 'Cinematic video', 'https://lumalabs.ai', 'h
('elevenlabs', 'elevenlabs', 'ElevenLabs', 'text_to_speech', 'Voice synthesis', 'https://ele
('deepgram', 'deepgram', 'Deepgram', 'speech_to_text', 'Speech recognition', 'https://deepgr
('perplexity', 'perplexity', 'Perplexity', 'search', 'AI search engine', 'https://perplexity
('voyage', 'voyage', 'Voyage AI', 'embedding', 'Premium embeddings', 'https://voyageai.com',
('meshy', 'meshy', 'Meshy', '3d_generation', 'AI 3D models', 'https://meshy.ai', 'https://ap

INSERT INTO schema_migrations (version, name, applied_by) VALUES ('007', 'external_providers
```

PART 4: DYNAMODB TABLES

packages/infrastructure/dynamodb/tables.ts

```typescript
/**
 * DynamoDB Table Definitions
 * For session, cache, and real-time data
 *
 * RADIANT v2.2.0 - December 2024
 */

import * as cdk from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import { Construct } from 'constructs';

// ============================================================================
// TABLE CONFIGURATIONS
// ============================================================================

export interface TableConfig {
  tableName: string;
  partitionKey: { name: string; type: dynamodb.AttributeType };
  sortKey?: { name: string; type: dynamodb.AttributeType };
  gsis?: GlobalSecondaryIndexConfig[];
  ttlAttribute?: string;
  stream?: dynamodb.StreamViewType;
  billingMode: dynamodb.BillingMode;
  pointInTimeRecovery: boolean;
}

export interface GlobalSecondaryIndexConfig {
  indexName: string;
```

```
  partitionKey: { name: string; type: dynamodb.AttributeType };
  sortKey?: { name: string; type: dynamodb.AttributeType };
  projectionType: dynamodb.ProjectionType;
}

// ============================================================================
// SESSIONS TABLE
// ============================================================================

export const SESSIONS_TABLE: TableConfig = {
  tableName: 'radiant-sessions',
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING },
  sortKey: { name: 'sk', type: dynamodb.AttributeType.STRING },
  gsis: [
    {
      indexName: 'gsi-session-token',
      partitionKey: { name: 'sessionToken', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.ALL,
    },
    {
      indexName: 'gsi-tenant-created',
      partitionKey: { name: 'tenantId', type: dynamodb.AttributeType.STRING },
      sortKey: { name: 'createdAt', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.KEYS_ONLY,
    },
  ],
  ttlAttribute: 'expiresAt',
  stream: dynamodb.StreamViewType.NEW_AND_OLD_IMAGES,
  billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
  pointInTimeRecovery: true,
};

// Session item structure:
// {
//   pk: 'tenant#<tenant-id>#user#<user-id>',
//   sk: 'session#<session-id>',
//   sessionToken: '<jwt-token-hash>',
//   tenantId: '<tenant-id>',
//   userId: '<user-id>',
//   createdAt: '<iso-timestamp>',
//   expiresAt: <unix-timestamp>,
//   lastActivityAt: '<iso-timestamp>',
//   ipAddress: '<ip>',
//   userAgent: '<user-agent>',
//   metadata: { ... }
// }
```

```typescript
// ============================================================================
// CHAT HISTORY TABLE
// ============================================================================

export const CHAT_HISTORY_TABLE: TableConfig = {
  tableName: 'radiant-chat-history',
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING },
  sortKey: { name: 'sk', type: dynamodb.AttributeType.STRING },
  gsis: [
    {
      indexName: 'gsi-conversation',
      partitionKey: { name: 'conversationId', type: dynamodb.AttributeType.STRING },
      sortKey: { name: 'createdAt', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.ALL,
    },
    {
      indexName: 'gsi-tenant-recent',
      partitionKey: { name: 'tenantId', type: dynamodb.AttributeType.STRING },
      sortKey: { name: 'createdAt', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.KEYS_ONLY,
    },
  ],
  ttlAttribute: 'expiresAt',
  stream: dynamodb.StreamViewType.NEW_IMAGE,
  billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
  pointInTimeRecovery: true,
};

// ============================================================================
// MODEL THERMAL STATE TABLE
// ============================================================================

export const MODEL_STATE_TABLE: TableConfig = {
  tableName: 'radiant-model-state',
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING },
  sortKey: { name: 'sk', type: dynamodb.AttributeType.STRING },
  gsis: [
    {
      indexName: 'gsi-thermal-state',
      partitionKey: { name: 'thermalState', type: dynamodb.AttributeType.STRING },
      sortKey: { name: 'lastRequestAt', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.ALL,
    },
  ],
  billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
```

```
  pointInTimeRecovery: true,
};

// ============================================================================
// CACHE TABLE
// ============================================================================

export const CACHE_TABLE: TableConfig = {
  tableName: 'radiant-cache',
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING },
  sortKey: { name: 'sk', type: dynamodb.AttributeType.STRING },
  ttlAttribute: 'expiresAt',
  billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
  pointInTimeRecovery: false,
};

// ============================================================================
// WEBSOCKET CONNECTIONS TABLE
// ============================================================================

export const CONNECTIONS_TABLE: TableConfig = {
  tableName: 'radiant-connections',
  partitionKey: { name: 'connectionId', type: dynamodb.AttributeType.STRING },
  gsis: [
    {
      indexName: 'gsi-tenant-user',
      partitionKey: { name: 'tenantId', type: dynamodb.AttributeType.STRING },
      sortKey: { name: 'userId', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.ALL,
    },
    {
      indexName: 'gsi-subscription',
      partitionKey: { name: 'subscription', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.ALL,
    },
  ],
  ttlAttribute: 'expiresAt',
  billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
  pointInTimeRecovery: false,
};

// ============================================================================
// APPROVAL QUEUE TABLE
// ============================================================================

export const APPROVAL_QUEUE_TABLE: TableConfig = {
```

```
  tableName: 'radiant-approval-queue',
  partitionKey: { name: 'pk', type: dynamodb.AttributeType.STRING },
  sortKey: { name: 'sk', type: dynamodb.AttributeType.STRING },
  gsis: [
    {
      indexName: 'gsi-approver',
      partitionKey: { name: 'approverPool', type: dynamodb.AttributeType.STRING },
      sortKey: { name: 'createdAt', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.ALL,
    },
    {
      indexName: 'gsi-initiator',
      partitionKey: { name: 'initiatedBy', type: dynamodb.AttributeType.STRING },
      sortKey: { name: 'createdAt', type: dynamodb.AttributeType.STRING },
      projectionType: dynamodb.ProjectionType.ALL,
    },
  ],
  ttlAttribute: 'expiresAt',
  stream: dynamodb.StreamViewType.NEW_AND_OLD_IMAGES,
  billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
  pointInTimeRecovery: true,
};

// ============================================================================
// TABLE FACTORY
// ============================================================================

export function createDynamoDBTable(
  scope: Construct,
  id: string,
  config: TableConfig,
  props: {
    resourcePrefix: string;
    environment: string;
    encryptionKey?: cdk.aws_kms.IKey;
    removalPolicy?: cdk.RemovalPolicy;
  }
): dynamodb.Table {
  const tableName = `${props.resourcePrefix}-${config.tableName}`;

  const table = new dynamodb.Table(scope, id, {
    tableName,
    partitionKey: config.partitionKey,
    sortKey: config.sortKey,
    billingMode: config.billingMode,
    pointInTimeRecovery: config.pointInTimeRecovery,
```

```
    encryption: props.encryptionKey
      ? dynamodb.TableEncryption.CUSTOMER_MANAGED
      : dynamodb.TableEncryption.AWS_MANAGED,
    encryptionKey: props.encryptionKey,
    removalPolicy: props.removalPolicy ?? cdk.RemovalPolicy.RETAIN,
    stream: config.stream,
    timeToLiveAttribute: config.ttlAttribute,
  });

  if (config.gsis) {
    for (const gsi of config.gsis) {
      table.addGlobalSecondaryIndex({
        indexName: gsi.indexName,
        partitionKey: gsi.partitionKey,
        sortKey: gsi.sortKey,
        projectionType: gsi.projectionType,
      });
    }
  }

  return table;
}

// ============================================================================
// ALL TABLE CONFIGS
// ============================================================================

export const ALL_TABLES: TableConfig[] = [
  SESSIONS_TABLE,
  CHAT_HISTORY_TABLE,
  MODEL_STATE_TABLE,
  CACHE_TABLE,
  CONNECTIONS_TABLE,
  APPROVAL_QUEUE_TABLE,
];
```

PART 5: MIGRATION RUNNER

packages/infrastructure/migrations/migrations.ts

```
/**
 * Database Migration Runner
 * RADIANT v2.2.0 - December 2024
 */
```

```typescript
import { Client } from 'pg';
import * as fs from 'fs';
import * as path from 'path';
import * as crypto from 'crypto';

interface Migration {
  version: string;
  name: string;
  filename: string;
  sql: string;
  checksum: string;
}

interface MigrationResult {
  version: string;
  name: string;
  success: boolean;
  duration: number;
  error?: string;
}

export class MigrationRunner {
  private client: Client;
  private migrationsDir: string;

  constructor(connectionString: string, migrationsDir?: string) {
    this.client = new Client({ connectionString });
    this.migrationsDir = migrationsDir ?? path.join(__dirname, '.');
  }

  async connect(): Promise<void> {
    await this.client.connect();
  }

  async disconnect(): Promise<void> {
    await this.client.end();
  }

  async loadMigrations(): Promise<Migration[]> {
    const files = fs.readdirSync(this.migrationsDir)
      .filter(f => f.endsWith('.sql'))
      .sort();

    const migrations: Migration[] = [];

    for (const filename of files) {
```

```
    const match = filename.match(/^(\d{3})_(.+)\.sql$/);
    if (!match) continue;

    const [, version, name] = match;
    const filepath = path.join(this.migrationsDir, filename);
    const sql = fs.readFileSync(filepath, 'utf-8');
    const checksum = crypto.createHash('sha256').update(sql).digest('hex');

    migrations.push({ version, name, filename, sql, checksum });
  }

  return migrations;
}

async getAppliedMigrations(): Promise<Map<string, { name: string; checksum: string }>> {
  const result = await this.client.query(`
    SELECT version, name, checksum FROM schema_migrations ORDER BY version
  `);

  const applied = new Map<string, { name: string; checksum: string }>();
  for (const row of result.rows) {
    applied.set(row.version, { name: row.name, checksum: row.checksum });
  }

  return applied;
}

async runMigrations(options: { dryRun?: boolean; force?: boolean } = {}): Promise<Migratio
  const results: MigrationResult[] = [];
  const migrations = await this.loadMigrations();
  const applied = await this.getAppliedMigrations();

  for (const migration of migrations) {
    if (applied.has(migration.version)) {
      const existing = applied.get(migration.version)!;
      if (existing.checksum && existing.checksum !== migration.checksum && !options.force)
        throw new Error(`Migration ${migration.version} checksum mismatch`);
      }
      continue;
    }

    const startTime = Date.now();

    try {
      if (options.dryRun) {
        console.log(`[DRY RUN] Would apply: ${migration.version} - ${migration.name}`);
```

```
      } else {
        console.log(`Applying: ${migration.version} - ${migration.name}...`);

        await this.client.query('BEGIN');
        await this.client.query(migration.sql);
        await this.client.query(`
          INSERT INTO schema_migrations (version, name, checksum, applied_by)
          VALUES ($1, $2, $3, $4)
          ON CONFLICT (version) DO UPDATE SET checksum = EXCLUDED.checksum
        `, [migration.version, migration.name, migration.checksum, 'migration-runner']);
        await this.client.query('COMMIT');
      }

      results.push({
        version: migration.version,
        name: migration.name,
        success: true,
        duration: Date.now() - startTime,
      });

    } catch (error) {
      await this.client.query('ROLLBACK');
      results.push({
        version: migration.version,
        name: migration.name,
        success: false,
        duration: Date.now() - startTime,
        error: error instanceof Error ? error.message : String(error),
      });
      throw error;
    }
  }

  return results;
}

async getStatus(): Promise<{
  applied: { version: string; name: string; appliedAt: string }[];
  pending: { version: string; name: string }[];
}> {
  const migrations = await this.loadMigrations();
  const applied = await this.getAppliedMigrations();

  const result = await this.client.query(`
    SELECT version, name, applied_at FROM schema_migrations ORDER BY version
  `);
```

```
    const pending = migrations
      .filter(m => !applied.has(m.version))
      .map(m => ({ version: m.version, name: m.name }));

    return {
      applied: result.rows.map(r => ({
        version: r.version,
        name: r.name,
        appliedAt: r.applied_at,
      })),
      pending,
    };
  }
}

// CLI
async function main() {
  const command = process.argv[2];
  const connectionString = process.env.DATABASE_URL;

  if (!connectionString) {
    console.error('DATABASE_URL environment variable is required');
    process.exit(1);
  }

  const runner = new MigrationRunner(connectionString);

  try {
    await runner.connect();

    switch (command) {
      case 'status': {
        const status = await runner.getStatus();
        console.log('\n=== Applied Migrations ===');
        for (const m of status.applied) {
          console.log(`  Ã¢Å"â€œ ${m.version} - ${m.name} (${m.appliedAt})`);
        }
        console.log('\n=== Pending Migrations ===');
        for (const m of status.pending) {
          console.log(`  Ã¢â€"â€¦ ${m.version} - ${m.name}`);
        }
        break;
      }

      case 'migrate': {
```

```
        const dryRun = process.argv.includes('--dry-run');
        const force = process.argv.includes('--force');
        const results = await runner.runMigrations({ dryRun, force });

        console.log('\n=== Migration Results ===');
        for (const r of results) {
          const status = r.success ? 'Ã¢Å"â€œ' : 'Ã¢Å"â€"';
          console.log(`  ${status} ${r.version} - ${r.name} (${r.duration}ms)`);
          if (r.error) console.log(`    Error: ${r.error}`);
        }
        break;
      }

      default:
        console.log('Usage: migrations.ts <command>');
        console.log('Commands: status, migrate');
        console.log('Options: --dry-run, --force');
    }

  } finally {
    await runner.disconnect();
  }
}

if (require.main === module) {
  main().catch(err => {
    console.error('Migration error:', err);
    process.exit(1);
  });
}
```

PART 6: ENHANCED PRICING METADATA SYSTEM (v4.12)

NEW in v4.12: Comprehensive pricing metadata for Brain cost optimization.

Terminology: - **COST** = What RADIANT pays to providers (internal, admin-visible only) - **PRICE** = What customers pay (cost × markup) - visible to clients/Think Tank

Key Behavior: - Metadata service syncs **costs** from providers during model registry updates - Brain optimizes on **PRICE** when users request cost optimization - Admin sees: Cost + Markup % + Price + Estimated flags - Client/Think Tank sees: Price only + Estimated

price flag

packages/infrastructure/lib/config/providers/pricing.config.ts

```
/**
 * Pricing Configuration - RADIANT v4.12
 *
 * TERMINOLOGY:
 *   COST  = What RADIANT pays (internal)
 *   PRICE = What customers pay (cost × markup)
 *
 * December 2024
 */

// =============================================================================
// MARKUP CONFIGURATION
// =============================================================================

export const DEFAULT_MARKUP_RATES = {
  EXTERNAL_PERCENT: 40,       // 40% markup on external providers
  SELF_HOSTED_PERCENT: 75,    // 75% markup on self-hosted models
};

export const MARKUP_RATES = {
  EXTERNAL: 1.40,             // Multiplier for external (1 + 40%)
  SELF_HOSTED: 1.75,          // Multiplier for self-hosted (1 + 75%)
};

// =============================================================================
// VOLUME DISCOUNTS
// =============================================================================

export const VOLUME_DISCOUNTS = [
  { minSpend: 100, discount: 0.05 },    // 5% off at $100/month
  { minSpend: 500, discount: 0.10 },    // 10% off at $500/month
  { minSpend: 1000, discount: 0.15 },   // 15% off at $1,000/month
  { minSpend: 5000, discount: 0.20 },   // 20% off at $5,000/month
  { minSpend: 10000, discount: 0.25 },  // 25% off at $10,000/month
];

// =============================================================================
// FREE TIER LIMITS (per month)
// =============================================================================

export const FREE_TIER = {
  tokens: 100_000,
```

```
  requests: 1_000,
  images: 10,
  videoSeconds: 60,
  audioMinutes: 10,
  embeddings: 10_000,
};

// ============================================================================
// TIER RATE LIMITS
// ============================================================================

export const TIER_LIMITS = {
  1: { rpm: 60, tpm: 100_000 },       // SEED
  2: { rpm: 200, tpm: 500_000 },      // STARTUP
  3: { rpm: 1000, tpm: 2_000_000 },   // GROWTH
  4: { rpm: 5000, tpm: 10_000_000 },  // SCALE
  5: { rpm: 20000, tpm: 50_000_000 }, // ENTERPRISE
};

// ============================================================================
// COST → PRICE CONVERSION
// ============================================================================

/**
 * Convert provider cost to customer price
 */
export function costToPrice(cost: number, markupPercent: number): number {
  return cost * (1 + markupPercent / 100);
}

/**
 * Convert customer price back to provider cost (admin reference)
 */
export function priceToCost(price: number, markupPercent: number): number {
  return price / (1 + markupPercent / 100);
}

/**
 * Calculate billed price from base cost
 */
export function calculateBilledPrice(
  baseCost: number,
  isSelfHosted: boolean
): number {
  const markupPercent = isSelfHosted
    ? DEFAULT_MARKUP_RATES.SELF_HOSTED_PERCENT
```

```
    : DEFAULT_MARKUP_RATES.EXTERNAL_PERCENT;
  return costToPrice(baseCost, markupPercent);
}

// Legacy alias for backward compatibility
export const calculateBilledCost = calculateBilledPrice;

export function applyVolumeDiscount(
  totalSpend: number,
  billedAmount: number
): number {
  let discount = 0;
  for (const tier of VOLUME_DISCOUNTS) {
    if (totalSpend >= tier.minSpend) {
      discount = tier.discount;
    }
  }
  return billedAmount * (1 - discount);
}

export function isWithinFreeTier(
  usage: {
    tokens?: number;
    requests?: number;
    images?: number;
    videoSeconds?: number;
    audioMinutes?: number;
  }
): boolean {
  if ((usage.tokens ?? 0) > FREE_TIER.tokens) return false;
  if ((usage.requests ?? 0) > FREE_TIER.requests) return false;
  if ((usage.images ?? 0) > FREE_TIER.images) return false;
  if ((usage.videoSeconds ?? 0) > FREE_TIER.videoSeconds) return false;
  if ((usage.audioMinutes ?? 0) > FREE_TIER.audioMinutes) return false;
  return true;
}
```

packages/infrastructure/lib/config/providers/cost-metadata.types.ts

```
/**
 * Cost Metadata Types - RADIANT v4.12
 *
 * Internal cost data collected by metadata service.
 * VISIBLE TO: Administrators only
 */
```

```typescript
// ============================================================================
// COST METADATA (Admin-Only)
// ============================================================================

export interface ModelCostMetadata {
  modelId: string;
  providerId: string;

  // Cost type
  costType: 'per_token' | 'per_request' | 'per_second' | 'per_image' | 'per_minute';

  // Base token costs (what RADIANT pays)
  baseCosts: {
    inputPer1kTokens: number;
    outputPer1kTokens: number;
  };

  // Context caching cost discounts
  cachingCosts: {
    supported: boolean;
    cachedInputPer1kTokens?: number;
    discountPercent?: number;           // e.g., 90 = 90% off
    minTokensForCaching?: number;
    cacheTTLSeconds?: number;
  };

  // Batch API cost discounts
  batchCosts: {
    supported: boolean;
    batchInputPer1kTokens?: number;
    batchOutputPer1kTokens?: number;
    discountPercent?: number;           // e.g., 50 = 50% off
    maxLatencyHours?: number;
  };

  // Long context costs
  longContextCosts?: {
    thresholdTokens: number;
    inputPer1kTokens: number;
    outputPer1kTokens: number;
  };

  // Special costs
  specialCosts: {
    perRequest?: number;
    perSecond?: number;
```

```
    perImage?: number;
    perMinute?: number;
    perSearchRequest?: number;
    perToolCall?: number;
  };

  // Source and freshness
  source: CostSource;
  lastVerifiedAt: string;
  lastUpdatedAt: string;
  nextSyncDueAt: string;

  // Estimated cost flag
  isEstimated: boolean;
  estimatedInfo?: EstimatedCostInfo;
}

export type CostSource =
  | 'provider_api'
  | 'provider_documentation'
  | 'admin_manual_entry'
  | 'registry_sync'
  | 'estimated_average'
  | 'historical_fallback';

export interface EstimatedCostInfo {
  reason: EstimatedCostReason;
  confidence: number;                    // 0.0 - 1.0

  sourceModels: Array<{
    modelId: string;
    displayName: string;
    providerId: string;
    inputCostPer1k: number;
    outputCostPer1k: number;
    similarityScore: number;
    weight: number;
  }>;

  calculationMethod: 'weighted_average' | 'simple_average' | 'nearest_model';
  similarityFactors: string[];
  estimatedAt: string;

  // Admin tracking
  adminNotified: boolean;
  adminNotifiedAt?: string;
```

```
  adminAdjusted: boolean;
  adminAdjustedBy?: string;
  adminAdjustedAt?: string;

  awaitingRealCost: boolean;
  expectedResolutionDate?: string;
}

export type EstimatedCostReason =
  | 'model_temporarily_unavailable'
  | 'cost_not_published'
  | 'new_model_pending'
  | 'provider_api_error'
  | 'provider_rate_limited'
  | 'deprecated_model'
  | 'preview_beta_model';

// ============================================================================
// THERMAL COST FACTORS (Self-Hosted)
// ============================================================================

export interface ThermalCostFactors {
  modelId: string;
  currentState: ThermalState;

  warmupCosts: {
    estimatedCost: number;
    estimatedDurationSeconds: number;
  };

  hourlyInfrastructureCosts: {
    OFF: number;
    COLD: number;
    WARM: number;
    HOT: number;
  };

  instanceDetails: {
    type: string;
    costPerHour: number;
  };

  perRequestCost: {
    amortizedCostPerRequest: number;
  };
}
```

```typescript
export type ThermalState = 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';
```

packages/infrastructure/lib/config/providers/pricing-views.ts

```typescript
/**
 * Role-Based Pricing Views - RADIANT v4.12
 *
 * Admin View:  Shows COST + MARKUP % + PRICE + Estimated flags
 * Client View: Shows PRICE only + Estimated price flag
 */

import { ModelCostMetadata, CostSource } from './cost-metadata.types';
import { costToPrice, DEFAULT_MARKUP_RATES } from './pricing.config';

// =============================================================================
// MARKUP CONFIGURATION
// =============================================================================

export interface MarkupConfiguration {
  defaults: {
    externalProvidersPercent: number;
    selfHostedModelsPercent: number;
  };

  providerOverrides: Record<string, {
    markupPercent: number;
    setBy: string;
    setAt: string;
    reason?: string;
  }>;

  modelOverrides: Record<string, {
    markupPercent: number;
    setBy: string;
    setAt: string;
    reason?: string;
    expiresAt?: string;
  }>;
}

export function getEffectiveMarkup(
  modelId: string,
  providerId: string,
  isExternal: boolean,
  config: MarkupConfiguration
```

```
): { percent: number; source: 'model' | 'provider' | 'default' } {
  // Priority: Model > Provider > Default
  const modelOverride = config.modelOverrides[modelId];
  if (modelOverride && (!modelOverride.expiresAt || new Date(modelOverride.expiresAt) > new
    return { percent: modelOverride.markupPercent, source: 'model' };
  }

  const providerOverride = config.providerOverrides[providerId];
  if (providerOverride) {
    return { percent: providerOverride.markupPercent, source: 'provider' };
  }

  return {
    percent: isExternal
      ? config.defaults.externalProvidersPercent
      : config.defaults.selfHostedModelsPercent,
    source: 'default'
  };
}

// ============================================================================
// ADMIN PRICING VIEW (Full Visibility)
// ============================================================================

export interface AdminPricingView {
  modelId: string;
  modelDisplayName: string;
  providerId: string;
  providerDisplayName: string;

  // Our COST (what RADIANT pays) - ADMIN ONLY
  cost: {
    inputPer1M: string;                 // "$3.00"
    outputPer1M: string;                // "$15.00"
    source: CostSource;
    lastUpdated: string;
    cachedInputPer1M?: string;
    batchInputPer1M?: string;
    batchOutputPer1M?: string;
  };

  // MARKUP configuration
  markup: {
    percent: number;
    source: 'model' | 'provider' | 'default';
    isOverridden: boolean;
```

```
    overriddenBy?: string;
    overriddenAt?: string;
  };

  // Customer PRICE (what they pay)
  price: {
    inputPer1M: string;                  // "$4.20"
    outputPer1M: string;                 // "$21.00"
    cachedInputPer1M?: string;
    batchInputPer1M?: string;
    batchOutputPer1M?: string;
  };

  // MARGIN (price - cost)
  margin: {
    inputPer1M: string;
    outputPer1M: string;
    percentOfPrice: number;
  };

  // ESTIMATED FLAG
  estimatedCost: {
    isEstimated: boolean;
    reason?: string;
    confidence?: number;
    sourceModels?: string[];
    adminCanAdjust: boolean;
    awaitingRealCost: boolean;
  };

  // Admin actions
  actions: {
    canOverrideMarkup: boolean;
    canAdjustEstimatedCost: boolean;
    canForceSync: boolean;
  };
}

// ============================================================================
// CLIENT PRICING VIEW (Price Only)
// ============================================================================

export interface ClientPricingView {
  modelId: string;
  modelDisplayName: string;
  providerDisplayName: string;
```

```
  // PRICES only (what customer pays) - NO COST/MARGIN VISIBLE
  prices: {
    inputPer1M: string;                 // "$4.20 / 1M tokens"
    outputPer1M: string;                // "$21.00 / 1M tokens"
  };

  // Savings options
  savingsOptions?: {
    caching?: {
      available: boolean;
      cachedInputPer1M: string;
      savingsPercent: number;
      description: string;
    };
    batch?: {
      available: boolean;
      batchInputPer1M: string;
      batchOutputPer1M: string;
      savingsPercent: number;
      description: string;
    };
  };

  // ESTIMATED PRICE FLAG (shown to customer)
  estimatedPrice: {
    isEstimated: boolean;
    badge?: {
      text: string;                     // "Estimated"
      tooltip: string;                  // "Price is estimated and may change"
      variant: 'warning' | 'info';
    };
  };

  // Price comparison
  priceTier: 'budget' | 'standard' | 'premium';
  comparedToAverage: 'cheaper' | 'average' | 'expensive';
}

// ============================================================================
// VIEW FORMATTERS
// ============================================================================

export function formatAdminPricingView(
  costMetadata: ModelCostMetadata,
  markupConfig: MarkupConfiguration,
```

```
  isExternal: boolean,
  modelDisplayName: string,
  providerDisplayName: string
): AdminPricingView {
  const markup = getEffectiveMarkup(
    costMetadata.modelId,
    costMetadata.providerId,
    isExternal,
    markupConfig
  );

  const formatMoney = (amount: number) => `$${(amount * 1000).toFixed(2)}`;
  const toPrice = (cost: number) => costToPrice(cost, markup.percent);

  const inputCost = costMetadata.baseCosts.inputPer1kTokens;
  const outputCost = costMetadata.baseCosts.outputPer1kTokens;
  const inputPrice = toPrice(inputCost);
  const outputPrice = toPrice(outputCost);

  return {
    modelId: costMetadata.modelId,
    modelDisplayName,
    providerId: costMetadata.providerId,
    providerDisplayName,

    cost: {
      inputPer1M: formatMoney(inputCost),
      outputPer1M: formatMoney(outputCost),
      source: costMetadata.source,
      lastUpdated: costMetadata.lastUpdatedAt,
      cachedInputPer1M: costMetadata.cachingCosts.cachedInputPer1kTokens
        ? formatMoney(costMetadata.cachingCosts.cachedInputPer1kTokens)
        : undefined,
      batchInputPer1M: costMetadata.batchCosts.batchInputPer1kTokens
        ? formatMoney(costMetadata.batchCosts.batchInputPer1kTokens)
        : undefined,
      batchOutputPer1M: costMetadata.batchCosts.batchOutputPer1kTokens
        ? formatMoney(costMetadata.batchCosts.batchOutputPer1kTokens)
        : undefined,
    },

    markup: {
      percent: markup.percent,
      source: markup.source,
      isOverridden: markup.source !== 'default',
    },
```

```
    price: {
      inputPer1M: formatMoney(inputPrice),
      outputPer1M: formatMoney(outputPrice),
      cachedInputPer1M: costMetadata.cachingCosts.cachedInputPer1kTokens
        ? formatMoney(toPrice(costMetadata.cachingCosts.cachedInputPer1kTokens))
        : undefined,
      batchInputPer1M: costMetadata.batchCosts.batchInputPer1kTokens
        ? formatMoney(toPrice(costMetadata.batchCosts.batchInputPer1kTokens))
        : undefined,
      batchOutputPer1M: costMetadata.batchCosts.batchOutputPer1kTokens
        ? formatMoney(toPrice(costMetadata.batchCosts.batchOutputPer1kTokens))
        : undefined,
    },

    margin: {
      inputPer1M: formatMoney(inputPrice - inputCost),
      outputPer1M: formatMoney(outputPrice - outputCost),
      percentOfPrice: ((inputPrice - inputCost) / inputPrice) * 100,
    },

    estimatedCost: {
      isEstimated: costMetadata.isEstimated,
      reason: costMetadata.estimatedInfo?.reason,
      confidence: costMetadata.estimatedInfo?.confidence,
      sourceModels: costMetadata.estimatedInfo?.sourceModels.map(m => m.displayName),
      adminCanAdjust: costMetadata.isEstimated,
      awaitingRealCost: costMetadata.estimatedInfo?.awaitingRealCost || false,
    },

    actions: {
      canOverrideMarkup: true,
      canAdjustEstimatedCost: costMetadata.isEstimated,
      canForceSync: true,
    },
  };
}

export function formatClientPricingView(
  costMetadata: ModelCostMetadata,
  markupConfig: MarkupConfiguration,
  isExternal: boolean,
  modelDisplayName: string,
  providerDisplayName: string,
  averagePrices: { input: number; output: number }
): ClientPricingView {
```

```
const markup = getEffectiveMarkup(
  costMetadata.modelId,
  costMetadata.providerId,
  isExternal,
  markupConfig
);

const formatPrice = (amount: number) => `$${(amount * 1000).toFixed(2)} / 1M tokens`;
const toPrice = (cost: number) => costToPrice(cost, markup.percent);

const inputPrice = toPrice(costMetadata.baseCosts.inputPer1kTokens);

// Determine price tier
let priceTier: 'budget' | 'standard' | 'premium';
let comparedToAverage: 'cheaper' | 'average' | 'expensive';

if (inputPrice < averagePrices.input * 0.7) {
  priceTier = 'budget';
  comparedToAverage = 'cheaper';
} else if (inputPrice > averagePrices.input * 1.3) {
  priceTier = 'premium';
  comparedToAverage = 'expensive';
} else {
  priceTier = 'standard';
  comparedToAverage = 'average';
}

return {
  modelId: costMetadata.modelId,
  modelDisplayName,
  providerDisplayName,

  // ONLY PRICES - NO COST OR MARGIN
  prices: {
    inputPer1M: formatPrice(inputPrice),
    outputPer1M: formatPrice(toPrice(costMetadata.baseCosts.outputPer1kTokens)),
  },

  savingsOptions: {
    caching: costMetadata.cachingCosts.supported ? {
      available: true,
      cachedInputPer1M: formatPrice(toPrice(costMetadata.cachingCosts.cachedInputPer1kToke
      savingsPercent: costMetadata.cachingCosts.discountPercent || 0,
      description: `${costMetadata.cachingCosts.discountPercent}% off with context caching
    } : undefined,
    batch: costMetadata.batchCosts.supported ? {
```

```
        available: true,
        batchInputPer1M: formatPrice(toPrice(costMetadata.batchCosts.batchInputPer1kTokens!)
        batchOutputPer1M: formatPrice(toPrice(costMetadata.batchCosts.batchOutputPer1kTokens
        savingsPercent: costMetadata.batchCosts.discountPercent || 0,
        description: `${costMetadata.batchCosts.discountPercent}% off with batch API (24h)`,
      } : undefined,
    },

    // ESTIMATED FLAG - Customer sees this but doesn't know it's from estimated COST
    estimatedPrice: {
      isEstimated: costMetadata.isEstimated,
      badge: costMetadata.isEstimated ? {
        text: 'Estimated',
        tooltip: getClientEstimatedPriceTooltip(costMetadata.estimatedInfo?.reason),
        variant: 'warning',
      } : undefined,
    },

    priceTier,
    comparedToAverage,
  };
}

function getClientEstimatedPriceTooltip(reason?: string): string {
  const messages: Record<string, string> = {
    'model_temporarily_unavailable': 'Price is estimated - final pricing coming soon',
    'cost_not_published': 'Price not yet confirmed by provider',
    'new_model_pending': 'New model - pricing being finalized',
    'provider_api_error': 'Price temporarily unavailable',
    'provider_rate_limited': 'Price temporarily unavailable',
    'deprecated_model': 'Legacy model - estimated pricing',
    'preview_beta_model': 'Preview model - pricing may change',
  };
  return messages[reason || ''] || 'Price is estimated and may change';
}
```

packages/infrastructure/lib/config/providers/brain-price-optimizer.ts

```
/**
 * RADIANT Brain Price Optimizer - v4.12
 *
 * When users request cost optimization, the Brain optimizes based on PRICE
 * (what the customer pays), not our internal cost.
 */

import { ModelCostMetadata, ThermalCostFactors } from './cost-metadata.types';
```

```typescript
import { getEffectiveMarkup, MarkupConfiguration } from './pricing-views';
import { costToPrice } from './pricing.config';

// ============================================================================
// BRAIN PRICE OPTIMIZATION REQUEST
// ============================================================================

export interface BrainPriceOptimizationRequest {
  request: {
    estimatedInputTokens: number;
    estimatedOutputTokens: number;
    canUseCaching: boolean;
    cachedTokenCount?: number;
    canUseBatch: boolean;
    isLongContext: boolean;
    requiresToolCalls: boolean;
    estimatedToolCalls?: number;
  };

  requiredCapabilities: string[];
  pricePreference: 'minimize' | 'balance' | 'ignore';
  minQualityScore?: number;
  includeEstimatedPricing: boolean;

  budgetConstraints?: {
    maxPricePerRequest?: number;        // Customer's max price
    remainingMonthlyBudget?: number;    // Customer's remaining budget
  };
}

// ============================================================================
// MODEL PRICE ANALYSIS (Brain uses this)
// ============================================================================

export interface ModelPriceAnalysis {
  modelId: string;
  providerId: string;
  displayName: string;
  capabilities: string[];
  qualityScore: number;

  // === CALCULATED PRICES (What customer pays) - Brain optimizes on these ===
  calculatedPrices: {
    standardPrice: number;
    withCachingPrice?: number;
    withBatchPrice?: number;
```

```
    thermalOverheadPrice?: number;      // Self-hosted warm-up (marked up)
    totalPrice: number;

    breakdown: {
      inputTokensPrice: number;
      outputTokensPrice: number;
      cachingDiscount: number;
      batchDiscount: number;
      thermalPrice: number;
      toolCallsPrice: number;
    };
  };

  // === ADMIN-ONLY COST INFO (Not used for optimization) ===
  adminCostInfo: {
    totalCost: number;
    markupPercent: number;
    marginAmount: number;
  };

  // Flags
  flags: {
    isEstimatedPrice: boolean;
    isAvailable: boolean;
    requiresWarmup: boolean;
    warmupWaitSeconds?: number;
  };

  // Rankings (based on PRICE)
  rankings: {
    priceRank: number;
    valueRank: number;
    isCheapest: boolean;
    isBestValue: boolean;
    isRecommended: boolean;
  };
}

// ============================================================================
// BRAIN OPTIMIZATION RESULT
// ============================================================================

export interface BrainPriceOptimizationResult {
  selectedModel: ModelPriceAnalysis;
  selectionReason: string;
  allCandidates: ModelPriceAnalysis[];
```

```
  // All in PRICE terms (what customer pays)
  summary: {
    selectedPrice: number;
    cheapestPrice: number;
    averagePrice: number;
    savingsVsAverage: number;
    savingsPercent: number;
  };

  // Warnings for client (price-focused)
  clientWarnings: Array<{
    type: 'estimated_price' | 'warmup_delay' | 'near_budget';
    message: string;
  }>;

  // Warnings for admin (cost/margin-focused)
  adminWarnings: Array<{
    type: 'low_margin' | 'estimated_cost' | 'high_volume_discount';
    message: string;
    modelId?: string;
  }>;
}

// ============================================================================
// PRICE OPTIMIZATION FUNCTION
// ============================================================================

export function optimizeForPrice(
  candidates: ModelPriceAnalysis[],
  request: BrainPriceOptimizationRequest
): BrainPriceOptimizationResult {
  // Filter by capabilities
  let eligible = candidates.filter(c =>
    request.requiredCapabilities.every(cap => c.capabilities.includes(cap))
  );

  // Filter by availability
  eligible = eligible.filter(c => c.flags.isAvailable);

  // Filter estimated if not allowed
  if (!request.includeEstimatedPricing) {
    eligible = eligible.filter(c => !c.flags.isEstimatedPrice);
  }

  // Filter by quality
```

```
if (request.minQualityScore) {
  eligible = eligible.filter(c => c.qualityScore >= request.minQualityScore!);
}

// Filter by budget (PRICE constraint)
if (request.budgetConstraints?.maxPricePerRequest) {
  eligible = eligible.filter(c =>
    c.calculatedPrices.totalPrice <= request.budgetConstraints!.maxPricePerRequest!
  );
}

// Sort by PRICE preference
if (request.pricePreference === 'minimize') {
  // Sort by lowest PRICE
  eligible.sort((a, b) => a.calculatedPrices.totalPrice - b.calculatedPrices.totalPrice);
} else if (request.pricePreference === 'balance') {
  // Sort by value (quality per dollar of PRICE)
  eligible.sort((a, b) =>
    (b.qualityScore / b.calculatedPrices.totalPrice) -
    (a.qualityScore / a.calculatedPrices.totalPrice)
  );
}
// If 'ignore', keep original quality-based order

const selected = eligible[0];
const prices = eligible.map(c => c.calculatedPrices.totalPrice);
const cheapestPrice = Math.min(...prices);
const averagePrice = prices.reduce((a, b) => a + b, 0) / prices.length;

// Build client warnings
const clientWarnings: BrainPriceOptimizationResult['clientWarnings'] = [];
if (selected.flags.isEstimatedPrice) {
  clientWarnings.push({
    type: 'estimated_price',
    message: 'Price is estimated and may change',
  });
}
if (selected.flags.requiresWarmup) {
  clientWarnings.push({
    type: 'warmup_delay',
    message: `Model requires ~${selected.flags.warmupWaitSeconds}s warm-up`,
  });
}

// Build admin warnings
const adminWarnings: BrainPriceOptimizationResult['adminWarnings'] = [];
```

```
  const marginPercent = (selected.adminCostInfo.marginAmount / selected.calculatedPrices.tot
  if (marginPercent < 20) {
    adminWarnings.push({
      type: 'low_margin',
      message: `Low margin (${marginPercent.toFixed(1)}%) on ${selected.displayName}`,
      modelId: selected.modelId,
    });
  }
  if (selected.flags.isEstimatedPrice) {
    adminWarnings.push({
      type: 'estimated_cost',
      message: `Cost for ${selected.displayName} is estimated - verify margin`,
      modelId: selected.modelId,
    });
  }

  return {
    selectedModel: selected,
    selectionReason: buildSelectionReason(selected, request),
    allCandidates: eligible,
    summary: {
      selectedPrice: selected.calculatedPrices.totalPrice,
      cheapestPrice,
      averagePrice,
      savingsVsAverage: averagePrice - selected.calculatedPrices.totalPrice,
      savingsPercent: ((averagePrice - selected.calculatedPrices.totalPrice) / averagePrice)
    },
    clientWarnings,
    adminWarnings,
  };
}

function buildSelectionReason(
  model: ModelPriceAnalysis,
  request: BrainPriceOptimizationRequest
): string {
  const reasons: string[] = [];

  if (model.rankings.isCheapest) {
    reasons.push('lowest price');
  } else if (model.rankings.isBestValue) {
    reasons.push('best value (quality per dollar)');
  }

  if (request.pricePreference === 'minimize') {
    reasons.push('optimized for minimum price');
```

```
  } else if (request.pricePreference === 'balance') {
    reasons.push('balanced price and quality');
  }

  if (model.flags.isEstimatedPrice) {
    reasons.push('note: estimated pricing');
  }

  return `Selected ${model.displayName}: ${reasons.join(', ')}`;
}

/**
 * Calculate prices for a model given request parameters
 */
export function calculateModelPrices(
  costMetadata: ModelCostMetadata,
  thermalCosts: ThermalCostFactors | null,
  markupConfig: MarkupConfiguration,
  isExternal: boolean,
  request: BrainPriceOptimizationRequest['request']
): ModelPriceAnalysis['calculatedPrices'] & { adminCostInfo: ModelPriceAnalysis['adminCostIn
  const markup = getEffectiveMarkup(
    costMetadata.modelId,
    costMetadata.providerId,
    isExternal,
    markupConfig
  );

  const toPrice = (cost: number) => costToPrice(cost, markup.percent);

  // Calculate base costs
  let inputCost = costMetadata.baseCosts.inputPer1kTokens * (request.estimatedInputTokens /
  let outputCost = costMetadata.baseCosts.outputPer1kTokens * (request.estimatedOutputTokens

  // Apply caching discount to cost
  let cachingDiscount = 0;
  if (request.canUseCaching && costMetadata.cachingCosts.supported && request.cachedTokenCou
    const cachedCost = costMetadata.cachingCosts.cachedInputPer1kTokens! * (request.cachedTo
    const standardCost = costMetadata.baseCosts.inputPer1kTokens * (request.cachedTokenCount
    cachingDiscount = standardCost - cachedCost;
    inputCost -= cachingDiscount;
  }

  // Apply batch discount to cost
  let batchDiscount = 0;
  if (request.canUseBatch && costMetadata.batchCosts.supported) {
```

```
    const discountPercent = costMetadata.batchCosts.discountPercent || 0;
    batchDiscount = (inputCost + outputCost) * (discountPercent / 100);
    inputCost *= (1 - discountPercent / 100);
    outputCost *= (1 - discountPercent / 100);
  }

  // Thermal cost for self-hosted
  let thermalCost = 0;
  if (thermalCosts && thermalCosts.currentState === 'COLD') {
    thermalCost = thermalCosts.warmupCosts.estimatedCost;
  }

  // Tool calls cost
  let toolCallsCost = 0;
  if (request.requiresToolCalls && costMetadata.specialCosts.perToolCall) {
    toolCallsCost = costMetadata.specialCosts.perToolCall * (request.estimatedToolCalls || 1
  }

  const totalCost = inputCost + outputCost + thermalCost + toolCallsCost;
  const totalPrice = toPrice(totalCost);

  return {
    standardPrice: toPrice(inputCost + outputCost),
    withCachingPrice: cachingDiscount > 0 ? toPrice(inputCost + outputCost) : undefined,
    withBatchPrice: batchDiscount > 0 ? toPrice(inputCost + outputCost) : undefined,
    thermalOverheadPrice: thermalCost > 0 ? toPrice(thermalCost) : undefined,
    totalPrice,

    breakdown: {
      inputTokensPrice: toPrice(inputCost),
      outputTokensPrice: toPrice(outputCost),
      cachingDiscount: toPrice(cachingDiscount),
      batchDiscount: toPrice(batchDiscount),
      thermalPrice: toPrice(thermalCost),
      toolCallsPrice: toPrice(toolCallsCost),
    },

    adminCostInfo: {
      totalCost,
      markupPercent: markup.percent,
      marginAmount: totalPrice - totalCost,
    },
  };
}
```

packages/infrastructure/lib/config/providers/estimated-cost-alerts.ts

```typescript
/**
 * Estimated Cost Alert System - RADIANT v4.12
 *
 * Alerts administrators when costs are estimated.
 * Admin can adjust estimated costs until real costs are found.
 */

import { ModelCostMetadata, EstimatedCostReason, CostSource } from './cost-metadata.types';

export interface EstimatedCostAlert {
  id: string;
  createdAt: string;
  severity: 'info' | 'warning' | 'critical';

  modelId: string;
  modelDisplayName: string;
  providerId: string;
  providerDisplayName: string;

  reason: EstimatedCostReason;
  estimatedCost: {
    inputPer1k: number;
    outputPer1k: number;
  };
  confidence: number;

  sourceModels: Array<{
    modelId: string;
    displayName: string;
    inputCostPer1k: number;
    outputCostPer1k: number;
    similarityScore: number;
  }>;

  // Impact on customer price
  affectedPrice: {
    inputPer1k: number;
    outputPer1k: number;
    markupPercent: number;
  };

  recommendation: string;
```

```
  // Status workflow
  status: 'pending' | 'acknowledged' | 'adjusted' | 'resolved';
  acknowledgedBy?: string;
  acknowledgedAt?: string;
  adjustedBy?: string;
  adjustedAt?: string;
  adjustedCost?: {
    inputPer1k: number;
    outputPer1k: number;
  };
  resolvedBy?: string;
  resolvedAt?: string;
  resolutionSource?: CostSource;

  notifications: Array<{
    adminId: string;
    adminEmail: string;
    channel: 'email' | 'slack' | 'dashboard';
    sentAt: string;
  }>;
}

export function createEstimatedCostAlert(
  costMetadata: ModelCostMetadata,
  affectedPrice: { inputPer1k: number; outputPer1k: number; markupPercent: number },
  modelDisplayName: string,
  providerDisplayName: string
): EstimatedCostAlert {
  const info = costMetadata.estimatedInfo!;

  // Determine severity based on confidence
  let severity: 'info' | 'warning' | 'critical';
  if (info.confidence > 0.8) severity = 'info';
  else if (info.confidence > 0.5) severity = 'warning';
  else severity = 'critical';

  return {
    id: `alert_${costMetadata.modelId}_${Date.now()}`,
    createdAt: new Date().toISOString(),
    severity,

    modelId: costMetadata.modelId,
    modelDisplayName,
    providerId: costMetadata.providerId,
    providerDisplayName,
```

```
    reason: info.reason,
    estimatedCost: {
      inputPer1k: costMetadata.baseCosts.inputPer1kTokens,
      outputPer1k: costMetadata.baseCosts.outputPer1kTokens,
    },
    confidence: info.confidence,

    sourceModels: info.sourceModels.map(m => ({
      modelId: m.modelId,
      displayName: m.displayName,
      inputCostPer1k: m.inputCostPer1k,
      outputCostPer1k: m.outputCostPer1k,
      similarityScore: m.similarityScore,
    })),

    affectedPrice,

    recommendation: buildRecommendation(info.confidence, severity),

    status: 'pending',
    notifications: [],
  };
}

function buildRecommendation(confidence: number, severity: string): string {
  if (severity === 'critical') {
    return 'Low confidence estimate. Recommend manually verifying cost with provider or adju
  } else if (severity === 'warning') {
    return 'Moderate confidence estimate. Consider verifying with provider documentation.';
  }
  return 'High confidence estimate based on similar models. Will auto-update when real cost
}

/**
 * Admin action: Adjust estimated cost
 */
export function adjustEstimatedCost(
  alert: EstimatedCostAlert,
  newCost: { inputPer1k: number; outputPer1k: number },
  adminId: string
): EstimatedCostAlert {
  return {
    ...alert,
    status: 'adjusted',
    adjustedBy: adminId,
    adjustedAt: new Date().toISOString(),
```

```
    adjustedCost: newCost,
  };
}

/**
 * Resolve alert when real cost is found
 */
export function resolveAlert(
  alert: EstimatedCostAlert,
  source: CostSource,
  adminId?: string
): EstimatedCostAlert {
  return {
    ...alert,
    status: 'resolved',
    resolvedBy: adminId || 'system',
    resolvedAt: new Date().toISOString(),
    resolutionSource: source,
  };
}
```

PART 7: ESTIMATED PRICES

External Provider Costs (Example Monthly Usage)

Provider	Model	1M Tokens	10M Tokens	100M Tokens
OpenAI	GPT-4o	$17.50	$175	$1,750
OpenAI	GPT-4o-mini	$1.05	$10.50	$105
Anthropic	Claude Opus 4	$126	$1,260	$12,600
Anthropic	Claude Sonnet 4	$25.20	$252	$2,520
Anthropic	Claude 3.5 Haiku	$6.72	$67.20	$672
Google	Gemini 2.0 Flash	$0.70	$7.00	$70
DeepSeek	DeepSeek Chat	$1.92	$19.20	$192
DeepSeek	DeepSeek R1	$3.84	$38.40	$384

All prices include 40% markup

Database Costs (Monthly)

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Aurora (Primary)	~$29	~$75	~$200	~$500	~$1,500
DynamoDB	~$5	~$15	~$50	~$150	~$500
ElastiCache	N/A	~$25	~$75	~$200	~$500
Database Total	**~$34**	**~$115**	**~$325**	**~$850**	**~$2,500**

PROVIDER SUMMARY

Text Generation (7 providers, 15+ models)

- **OpenAI**: GPT-4o, GPT-4o-mini, O1, O1-mini, O3-mini
- **Anthropic**: Claude Opus 4, Sonnet 4, Haiku 3.5
- **Google**: Gemini 2.0 Flash, 1.5 Pro
- **xAI**: Grok 3, Grok 3 Mini
- **DeepSeek**: Chat, Reasoner (R1)
- **Mistral**: Large, Small, Codestral
- **Cohere**: Command R+, Command R

Image Generation (5 providers, 10+ models)

- **OpenAI**: DALL-E 3, DALL-E 3 HD
- **Stability**: SDXL, SD3 Large, Ultra
- **FLUX**: Pro 1.1, Pro Ultra, Schnell
- **Ideogram**: V2, V2 Turbo
- **Midjourney**: V6

Video Generation (5 providers, 7+ models)

- **Runway**: Gen-3 Alpha, Turbo
- **Luma**: Dream Machine, Ray 2
- **Pika**: Pika 2.0
- **Kling**: V1.5 Pro
- **Google Veo**: Veo 2

Audio (4 providers, 7+ models)

- **OpenAI**: TTS-1, TTS-1-HD, Whisper
- **ElevenLabs**: Multilingual V2, Turbo V2.5
- **Deepgram**: Nova-2, Nova-2 Medical
- **AssemblyAI**: Best

Embeddings (2 providers, 5 models)

- **OpenAI**: text-embedding-3-small/large
- **Voyage**: voyage-3, voyage-code-3

Search (3 providers)

- **Perplexity**: Sonar, Sonar Pro
- **Exa**: Neural Search
- **Tavily**: Search API

3D Generation (2 providers)

- **Meshy**: V3, V3 Turbo
- **Tripo**: V2

Total: 21 external providers, 50+ models

NEXT PROMPTS

Continue with: - **Prompt 8**: Admin Web Dashboard (Next.js) - **Prompt 9**: Assembly & Deployment Guide

---

End of Prompt 7: External Providers & Database Schema/Migrations RADIANT v2.2.0 - December 2024

â• â•

END OF SECTION 7

â• â•

â• â•

SECTION-08-ADMIN-DASHBOARD

SECTION 8: ADMIN WEB DASHBOARD (v2.2.0)

â•

> **Dependencies:** Sections 0-7 **Creates:** Next.js 14 dashboard, all management UIs

âš ï¸ IMPORTANT: Type Imports

The admin dashboard should import types from @radiant/shared, NOT redefine them.

lib/api/types.ts

```
/**
 * Re-export types from @radiant/shared for dashboard use
 * DO NOT redefine types here
 */

export {
  // App types
  type ManagedApp,
  type AppStatus,
  type EnvironmentStatus,
  type DeploymentStatus,

  // Environment types
  type Environment,
  type TierLevel,
  type TierConfig,

  // AI types
  type AIProvider,
  type AIModel,
  type ThermalState,
  type ServiceState,
  type ModelStatus,

  // Admin types
  type Administrator,
  type AdminRole,
  type AdminStatus,
```

```
  type Invitation,
  type ApprovalRequest,

  // Billing types
  type UsageEvent,
  type Invoice,
  type BillingBreakdown,

  // Constants
  TIER_CONFIGS,
  ENVIRONMENTS,
  EXTERNAL_PROVIDERS,
  ROLE_PERMISSIONS,

  // Utils
  formatCurrency,
  formatTokens,
  formatDuration,
} from '@radiant/shared';

// Dashboard-specific types (not in shared)
export interface DashboardMetrics {
  totalRequests: number;
  totalTokens: number;
  totalCost: number;
  activeModels: number;
  activeProviders: number;
  period: { start: Date; end: Date };
}

export interface SystemHealth {
  api: HealthStatus;
  database: HealthStatus;
  litellm: HealthStatus;
  sagemaker?: HealthStatus;
}

export type HealthStatus = 'healthy' | 'degraded' | 'unhealthy';
```

RADIANT v2.2.0 - Prompt 8: Admin Web Dashboard (Next.js)

Prompt 8 of 9 | Target Size: ~85KB | Version: 3.7.0 | December 2024

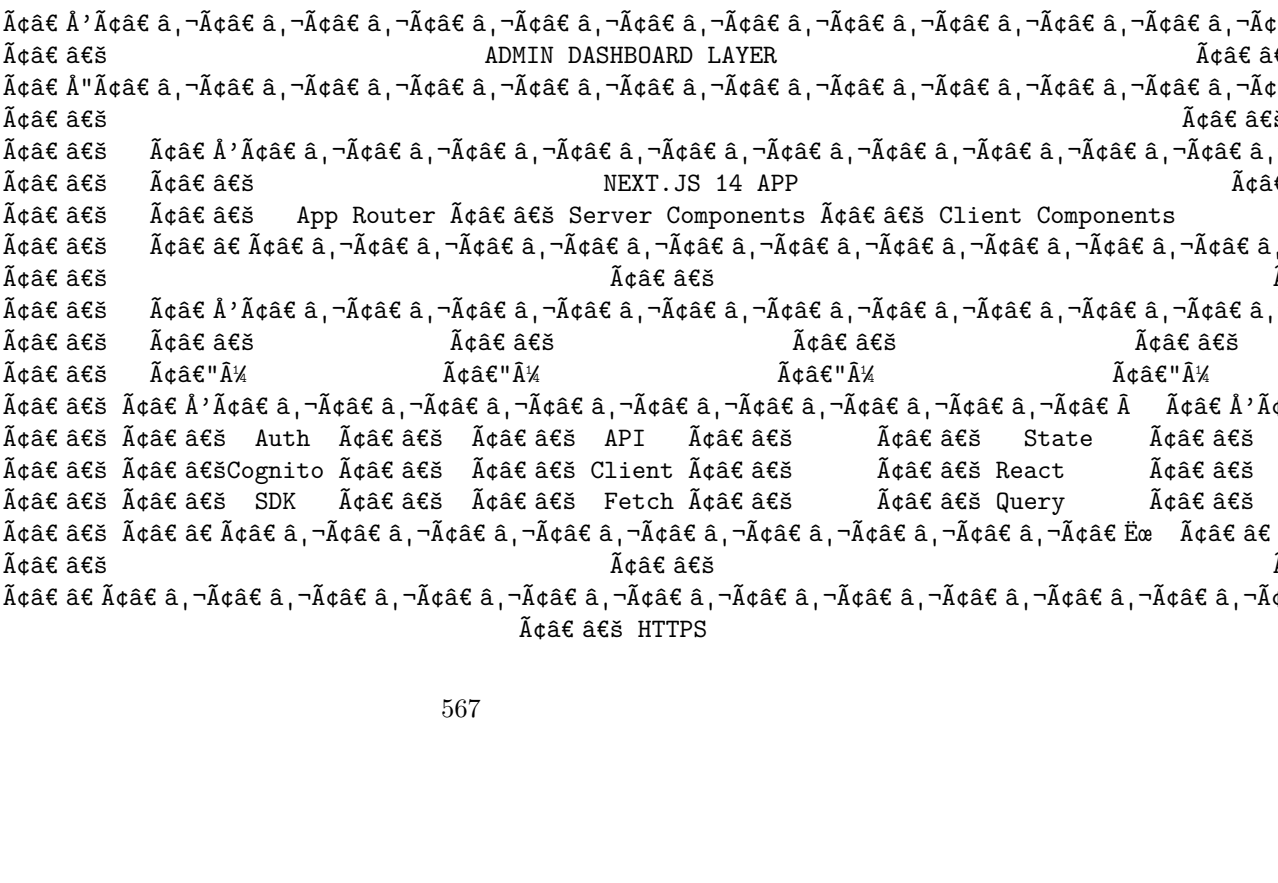
OVERVIEW

This prompt creates the complete Admin Web Dashboard for the RADIANT platform:

1. **Next.js 14 Project** - App Router, TypeScript, Tailwind CSS
2. **Authentication** - Cognito integration with admin pool
3. **Dashboard Pages** - Complete admin interface for platform management
4. **Component Library** - shadcn/ui with custom RADIANT components
5. **API Integration** - React Query with type-safe API client
6. **Real-time Features** - WebSocket notifications for model warm-up

The admin dashboard provides a complete web interface for managing the RADIANT platform, including AI model configuration, administrator management, billing, and two-person approval workflows.

ARCHITECTURE



```
                              Ã¢â€"Â¼
Ã¢â€ Å'Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
Ã¢â€ â€š                          API GATEWAY                          Ã¢â€ â
Ã¢â€ â€š                REST API (/api/v2/*) Ã¢â€ â€š AppSync GraphQL                Ã
Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢â€ â‚¬Ã¢
```

TECHNOLOGY STACK

Layer	Technology	Version	Purpose
Framework	Next.js	14.x	React framework with App Router
UI Library	React	18.x	Component library
Styling	Tailwind CSS	3.4.x	Utility-first CSS
Components	shadcn/ui	Latest	Accessible component primitives
Charts	Recharts	2.x	Data visualization
Icons	Lucide React	Latest	Icon library
State	React Query	5.x	Server state management
Forms	React Hook Form	7.x	Form handling
Validation	Zod	3.x	Schema validation
Date	date-fns	3.x	Date manipulation
Maps	react-simple-maps	3.x	Geographic visualization
Auth	AWS Amplify	6.x	Cognito authentication
Themes	next-themes	Latest	Dark mode support

PROJECT STRUCTURE

```
apps/admin-dashboard/
Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ app/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ (auth)/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ login/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ page.tsx              # Login page
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ forgot-password/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ page.tsx              # Password rese
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ accept-invitation/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ page.tsx              # Invitation ac
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ layout.tsx             # Auth layout (centere
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ (dashboard)/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ layout.tsx             # Dashboard layout wit
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ page.tsx               # Main dashboard
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ geographic/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ page.tsx              # Geographic ov
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ models/
```

```
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # AI models lis
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ [id]/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx         # Model detail/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ services/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # Mid-level ser
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ providers/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # External provi
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ [id]/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx         # Provider conf
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ administrators/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # Admin users li
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ invitations/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx         # Pendin
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ approvals/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx         # Two-person ap
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ billing/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # Billing overv
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ margins/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx         # Margin
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ invoices/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx         # Invoice histo
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ usage/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # Usage analyti
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ notifications/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # Notification
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ settings/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx             # General settings
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ profile/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx         # Admin profile
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ security/
Ã¢â€ â€š   Ã¢â€ â€š           Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ page.tsx          # MFA & security setti
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ api/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ [...path]/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ route.ts              # API proxy (BFF patt
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ globals.css                  # Design system tokens + Tai
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ layout.tsx                   # Root layout with providers
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ not-found.tsx                # 404 page
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ components/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ ui/                           # shadcn/ui base components
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ button.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ card.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ input.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ select.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ table.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ dialog.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ dropdown-menu.tsx
```

```
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ tabs.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ toast.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ badge.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ progress.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ tooltip.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ switch.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ slider.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ alert.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ avatar.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ command.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ sheet.tsx
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€¢Ã¢â€ â¸¬Ã¢â€ â¸¬ skeleton.tsx
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ layout/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ sidebar.tsx              # Main navigation side
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ header.tsx               # Top header bar
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ page-header.tsx          # Page title + breadcr
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€¢Ã¢â€ â¸¬Ã¢â€ â¸¬ mobile-nav.tsx          # Mobile navigation
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ dashboard/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ metric-card.tsx          # KPI display cards
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ activity-feed.tsx        # Recent activity
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ quick-actions.tsx        # Common actions
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€¢Ã¢â€ â¸¬Ã¢â€ â¸¬ system-health.tsx       # Health overview
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ geographic/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ world-map.tsx           # Interactive world ma
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ region-card.tsx          # Region statistics
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€¢Ã¢â€ â¸¬Ã¢â€ â¸¬ latency-heatmap.tsx     # Latency visualizati
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ models/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ model-card.tsx           # Model status card
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ thermal-badge.tsx        # Thermal state indica
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ thermal-controls.tsx     # Thermal state manage
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ model-filters.tsx        # Filter/search UI
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ model-table.tsx          # Models data table
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€¢Ã¢â€ â¸¬Ã¢â€ â¸¬ pricing-form.tsx        # Price configuration
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ services/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ service-card.tsx         # Service status card
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ service-state-badge.tsx  # Service state indica
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ dependency-tree.tsx      # Model dependencies
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€¢Ã¢â€ â¸¬Ã¢â€ â¸¬ service-controls.tsx     # Service management
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ providers/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ provider-card.tsx        # Provider status
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ api-key-input.tsx       # Masked API key input
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€¢Ã¢â€ â¸¬Ã¢â€ â¸¬ health-status.tsx       # Provider health che
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ administrators/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ admin-table.tsx          # Administrators list
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ invitation-form.tsx      # Send invitation moda
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ role-badge.tsx           # Role display
```

```
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ approval-card.tsx         # Approval request ca
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ billing/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ cost-chart.tsx            # Cost over time
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ margin-slider.tsx         # Margin configuration
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ usage-breakdown.tsx       # Usage by category
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ invoice-table.tsx         # Invoice list
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ charts/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ area-chart.tsx            # Time series chart
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ bar-chart.tsx             # Comparison chart
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ donut-chart.tsx           # Distribution chart
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ sparkline.tsx             # Mini inline chart
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ heatmap.tsx              # 2D visualization
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ notifications/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ notification-bell.tsx     # Header notification
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ notification-list.tsx     # Notification dropdow
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ toast-notifications.tsx   # Toast container
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ common/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ loading-spinner.tsx       # Loading states
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ empty-state.tsx           # No data display
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ error-boundary.tsx        # Error handling
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ confirm-dialog.tsx        # Confirmation modal
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ lib/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ api/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ client.ts                # API client with auth
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ endpoints.ts             # API endpoint definit
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ types.ts                # API response types
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ auth/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ amplify.ts               # Amplify configuratio
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ context.tsx               # Auth context provide
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ hooks.ts                # useAuth, useCurrent
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ hooks/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ use-models.ts             # Models queries
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ use-providers.ts          # Providers queries
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ use-administrators.ts     # Admin queries
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ use-billing.ts            # Billing queries
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ use-notifications.ts      # Notification queries
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ utils/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ cn.ts                    # Class name helper
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ format.ts                # Number/date formatti
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ constants.ts             # App constants
Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ websocket/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ client.ts                # WebSocket client
Ã¢â€ â€š       Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ hooks.ts                 # useWebSocket hook
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ public/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ world-110m.json             # World map topology
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ favicon.ico
```

```
Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ logo.svg
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ next.config.js
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ tailwind.config.ts
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ tsconfig.json
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ package.json
Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ .env.local.example
```

PART 1: PROJECT CONFIGURATION

package.json

```
{
  "name": "@radiant/admin-dashboard",
  "version": "2.2.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "type-check": "tsc --noEmit"
  },
  "dependencies": {
    "next": "^14.2.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",

    "@radix-ui/react-alert-dialog": "^1.0.5",
    "@radix-ui/react-avatar": "^1.0.4",
    "@radix-ui/react-checkbox": "^1.0.4",
    "@radix-ui/react-dialog": "^1.0.5",
    "@radix-ui/react-dropdown-menu": "^2.0.6",
    "@radix-ui/react-label": "^2.0.2",
    "@radix-ui/react-popover": "^1.0.7",
    "@radix-ui/react-progress": "^1.0.3",
    "@radix-ui/react-scroll-area": "^1.0.5",
    "@radix-ui/react-select": "^2.0.0",
    "@radix-ui/react-separator": "^1.0.3",
    "@radix-ui/react-slider": "^1.1.2",
    "@radix-ui/react-slot": "^1.0.2",
    "@radix-ui/react-switch": "^1.0.3",
    "@radix-ui/react-tabs": "^1.0.4",
    "@radix-ui/react-toast": "^1.1.5",
    "@radix-ui/react-tooltip": "^1.0.7",
```

```
    "@tanstack/react-query": "^5.24.0",
    "@tanstack/react-query-devtools": "^5.24.0",

    "aws-amplify": "^6.0.0",
    "@aws-amplify/adapter-nextjs": "^1.0.0",

    "react-hook-form": "^7.50.0",
    "@hookform/resolvers": "^3.3.4",
    "zod": "^3.22.4",

    "recharts": "^2.12.0",
    "react-simple-maps": "^3.0.0",
    "d3-geo": "^3.1.0",
    "topojson-client": "^3.1.0",

    "lucide-react": "^0.338.0",
    "class-variance-authority": "^0.7.0",
    "clsx": "^2.1.0",
    "tailwind-merge": "^2.2.0",

    "date-fns": "^3.3.0",
    "next-themes": "^0.2.1",
    "sonner": "^1.4.0",
    "cmdk": "^0.2.1"
  },
  "devDependencies": {
    "@types/node": "^20.11.0",
    "@types/react": "^18.2.0",
    "@types/react-dom": "^18.2.0",
    "@types/d3-geo": "^3.1.0",
    "@types/topojson-client": "^3.1.4",
    "autoprefixer": "^10.4.17",
    "postcss": "^8.4.35",
    "tailwindcss": "^3.4.1",
    "typescript": "^5.3.0",
    "eslint": "^8.56.0",
    "eslint-config-next": "^14.2.0"
  }
}
```

next.config.js

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  output: 'standalone',
```

```
// Environment variables exposed to browser
env: {
  NEXT_PUBLIC_API_URL: process.env.NEXT_PUBLIC_API_URL,
  NEXT_PUBLIC_COGNITO_REGION: process.env.NEXT_PUBLIC_COGNITO_REGION,
  NEXT_PUBLIC_COGNITO_USER_POOL_ID: process.env.NEXT_PUBLIC_COGNITO_USER_POOL_ID,
  NEXT_PUBLIC_COGNITO_CLIENT_ID: process.env.NEXT_PUBLIC_COGNITO_CLIENT_ID,
  NEXT_PUBLIC_WS_URL: process.env.NEXT_PUBLIC_WS_URL,
},

// Optimize images from S3/CloudFront
images: {
  remotePatterns: [
    {
      protocol: 'https',
      hostname: '*.cloudfront.net',
    },
    {
      protocol: 'https',
      hostname: '*.amazonaws.com',
    },
  ],
},

// Enable React strict mode
reactStrictMode: true,

// Disable x-powered-by header
poweredByHeader: false,

// Security headers
async headers() {
  return [
    {
      source: '/:path*',
      headers: [
        { key: 'X-Frame-Options', value: 'DENY' },
        { key: 'X-Content-Type-Options', value: 'nosniff' },
        { key: 'X-XSS-Protection', value: '1; mode=block' },
        { key: 'Referrer-Policy', value: 'strict-origin-when-cross-origin' },
        {
          key: 'Content-Security-Policy',
          value: [
            "default-src 'self'",
            "script-src 'self' 'unsafe-eval' 'unsafe-inline'",
            "style-src 'self' 'unsafe-inline'",
            "img-src 'self' data: https:",
```

```
              "font-src 'self'",
              `connect-src 'self' https://*.amazonaws.com https://*.cloudfront.net wss://*.a
            ].join('; '),
          },
        ],
      },
    ];
  },
};

module.exports = nextConfig;
```

tailwind.config.ts

```
import type { Config } from 'tailwindcss';

const config: Config = {
  darkMode: ['class'],
  content: [
    './app/**/*.{js,ts,jsx,tsx,mdx}',
    './components/**/*.{js,ts,jsx,tsx,mdx}',
    './lib/**/*.{js,ts,jsx,tsx,mdx}',
  ],
  theme: {
    container: {
      center: true,
      padding: '2rem',
      screens: {
        '2xl': '1400px',
      },
    },
    extend: {
      colors: {
        border: 'hsl(var(--border))',
        input: 'hsl(var(--input))',
        ring: 'hsl(var(--ring))',
        background: 'hsl(var(--background))',
        foreground: 'hsl(var(--foreground))',
        primary: {
          DEFAULT: 'hsl(var(--primary))',
          foreground: 'hsl(var(--primary-foreground))',
        },
        secondary: {
          DEFAULT: 'hsl(var(--secondary))',
          foreground: 'hsl(var(--secondary-foreground))',
        },
```

```
destructive: {
  DEFAULT: 'hsl(var(--destructive))',
  foreground: 'hsl(var(--destructive-foreground))',
},
muted: {
  DEFAULT: 'hsl(var(--muted))',
  foreground: 'hsl(var(--muted-foreground))',
},
accent: {
  DEFAULT: 'hsl(var(--accent))',
  foreground: 'hsl(var(--accent-foreground))',
},
popover: {
  DEFAULT: 'hsl(var(--popover))',
  foreground: 'hsl(var(--popover-foreground))',
},
card: {
  DEFAULT: 'hsl(var(--card))',
  foreground: 'hsl(var(--card-foreground))',
},
sidebar: {
  DEFAULT: 'hsl(var(--sidebar-background))',
  foreground: 'hsl(var(--sidebar-foreground))',
  border: 'hsl(var(--sidebar-border))',
  accent: 'hsl(var(--sidebar-accent))',
  'accent-foreground': 'hsl(var(--sidebar-accent-foreground))',
},
// Thermal states
thermal: {
  off: '#6b7280',       // gray-500
  cold: '#3b82f6',      // blue-500
  warm: '#f59e0b',      // amber-500
  hot: '#ef4444',       // red-500
  automatic: '#8b5cf6', // violet-500
},
// Service states
service: {
  running: '#22c55e',   // green-500
  degraded: '#f59e0b',  // amber-500
  disabled: '#6b7280',  // gray-500
  offline: '#ef4444',   // red-500
},
// Chart colors
chart: {
  1: 'hsl(var(--chart-1))',
  2: 'hsl(var(--chart-2))',
```

```
      3: 'hsl(var(--chart-3))',
      4: 'hsl(var(--chart-4))',
      5: 'hsl(var(--chart-5))',
    },
  },
  borderRadius: {
    lg: 'var(--radius)',
    md: 'calc(var(--radius) - 2px)',
    sm: 'calc(var(--radius) - 4px)',
  },
  fontFamily: {
    sans: ['Inter', 'system-ui', 'sans-serif'],
    mono: ['JetBrains Mono', 'monospace'],
  },
  keyframes: {
    'accordion-down': {
      from: { height: '0' },
      to: { height: 'var(--radix-accordion-content-height)' },
    },
    'accordion-up': {
      from: { height: 'var(--radix-accordion-content-height)' },
      to: { height: '0' },
    },
    'fade-in': {
      from: { opacity: '0' },
      to: { opacity: '1' },
    },
    'slide-in-from-right': {
      from: { transform: 'translateX(100%)', opacity: '0' },
      to: { transform: 'translateX(0)', opacity: '1' },
    },
    'pulse-slow': {
      '0%, 100%': { opacity: '1' },
      '50%': { opacity: '0.5' },
    },
    shimmer: {
      '0%': { backgroundPosition: '-200% 0' },
      '100%': { backgroundPosition: '200% 0' },
    },
  },
  animation: {
    'accordion-down': 'accordion-down 0.2s ease-out',
    'accordion-up': 'accordion-up 0.2s ease-out',
    'fade-in': 'fade-in 0.2s ease-out',
    'slide-in-from-right': 'slide-in-from-right 0.3s ease-out',
    'pulse-slow': 'pulse-slow 2s ease-in-out infinite',
```

```
        shimmer: 'shimmer 2s linear infinite',
      },
    },
  },
  plugins: [require('tailwindcss-animate')],
};

export default config;
```

app/globals.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

/* ============================================================================
   RADIANT ADMIN DESIGN SYSTEM
   Version: 3.7.0
   ============================================================================ */

@layer base {
  :root {
    /* Base Colors */
    --background: 0 0% 100%;
    --foreground: 222.2 84% 4.9%;

    /* Card */
    --card: 0 0% 100%;
    --card-foreground: 222.2 84% 4.9%;

    /* Popover */
    --popover: 0 0% 100%;
    --popover-foreground: 222.2 84% 4.9%;

    /* Primary - RADIANT Purple */
    --primary: 262 83% 58%;
    --primary-foreground: 210 40% 98%;

    /* Secondary */
    --secondary: 210 40% 96.1%;
    --secondary-foreground: 222.2 47.4% 11.2%;

    /* Muted */
    --muted: 210 40% 96.1%;
    --muted-foreground: 215.4 16.3% 46.9%;
```

```
  /* Accent */
  --accent: 210 40% 96.1%;
  --accent-foreground: 222.2 47.4% 11.2%;

  /* Destructive */
  --destructive: 0 84.2% 60.2%;
  --destructive-foreground: 210 40% 98%;

  /* Border & Input */
  --border: 214.3 31.8% 91.4%;
  --input: 214.3 31.8% 91.4%;
  --ring: 262 83% 58%;

  /* Radius */
  --radius: 0.5rem;

  /* Sidebar */
  --sidebar-background: 0 0% 98%;
  --sidebar-foreground: 240 5.3% 26.1%;
  --sidebar-border: 220 13% 91%;
  --sidebar-accent: 240 4.8% 95.9%;
  --sidebar-accent-foreground: 240 5.9% 10%;

  /* Charts */
  --chart-1: 262 83% 58%;
  --chart-2: 173 80% 40%;
  --chart-3: 197 37% 24%;
  --chart-4: 43 74% 66%;
  --chart-5: 27 87% 67%;
}

.dark {
  /* Base Colors */
  --background: 222.2 84% 4.9%;
  --foreground: 210 40% 98%;

  /* Card */
  --card: 222.2 84% 4.9%;
  --card-foreground: 210 40% 98%;

  /* Popover */
  --popover: 222.2 84% 4.9%;
  --popover-foreground: 210 40% 98%;

  /* Primary - RADIANT Purple */
  --primary: 262 83% 68%;
```

```
    --primary-foreground: 222.2 47.4% 11.2%;

    /* Secondary */
    --secondary: 217.2 32.6% 17.5%;
    --secondary-foreground: 210 40% 98%;

    /* Muted */
    --muted: 217.2 32.6% 17.5%;
    --muted-foreground: 215 20.2% 65.1%;

    /* Accent */
    --accent: 217.2 32.6% 17.5%;
    --accent-foreground: 210 40% 98%;

    /* Destructive */
    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 210 40% 98%;

    /* Border & Input */
    --border: 217.2 32.6% 17.5%;
    --input: 217.2 32.6% 17.5%;
    --ring: 262 83% 68%;

    /* Sidebar */
    --sidebar-background: 222.2 84% 6%;
    --sidebar-foreground: 210 40% 90%;
    --sidebar-border: 217.2 32.6% 17.5%;
    --sidebar-accent: 217.2 32.6% 15%;
    --sidebar-accent-foreground: 210 40% 98%;

    /* Charts */
    --chart-1: 262 83% 68%;
    --chart-2: 173 80% 50%;
    --chart-3: 197 37% 50%;
    --chart-4: 43 74% 66%;
    --chart-5: 27 87% 67%;
  }
}

@layer base {
  * {
    @apply border-border;
  }

  body {
    @apply bg-background text-foreground;
```

```
    font-feature-settings: "rlig" 1, "calt" 1;
  }

  /* Smooth scrolling */
  html {
    scroll-behavior: smooth;
  }

  /* Custom scrollbar */
  ::-webkit-scrollbar {
    width: 8px;
    height: 8px;
  }

  ::-webkit-scrollbar-track {
    @apply bg-muted rounded-full;
  }

  ::-webkit-scrollbar-thumb {
    @apply bg-muted-foreground/30 rounded-full hover:bg-muted-foreground/50;
  }
}

@layer components {
  /* Loading shimmer effect */
  .shimmer {
    background: linear-gradient(
      90deg,
      hsl(var(--muted)) 25%,
      hsl(var(--muted-foreground) / 0.1) 50%,
      hsl(var(--muted)) 75%
    );
    background-size: 200% 100%;
    animation: shimmer 2s linear infinite;
  }

  /* Focus ring */
  .focus-ring {
    @apply outline-none ring-2 ring-ring ring-offset-2 ring-offset-background;
  }
}

@layer utilities {
  /* Hide scrollbar but keep functionality */
  .scrollbar-hide {
    -ms-overflow-style: none;
```

```
    scrollbar-width: none;
  }

  .scrollbar-hide::-webkit-scrollbar {
    display: none;
  }
}
```

.env.local.example

```
# API Configuration
NEXT_PUBLIC_API_URL=https://api.your-domain.com
NEXT_PUBLIC_WS_URL=wss://ws.your-domain.com

# Cognito Configuration (Admin Pool)
NEXT_PUBLIC_COGNITO_REGION=us-east-1
NEXT_PUBLIC_COGNITO_USER_POOL_ID=us-east-1_XXXXXXXXX
NEXT_PUBLIC_COGNITO_CLIENT_ID=xxxxxxxxxxxxxxxxxxxxxxxxxx

# Environment
NEXT_PUBLIC_ENVIRONMENT=development
```

PART 2: AUTHENTICATION

lib/auth/amplify.ts

```
/**
 * AWS Amplify Configuration for RADIANT Admin Dashboard
 * Connects to the admin-specific Cognito user pool
 */

import { Amplify } from 'aws-amplify';

export function configureAmplify() {
  Amplify.configure({
    Auth: {
      Cognito: {
        userPoolId: process.env.NEXT_PUBLIC_COGNITO_USER_POOL_ID!,
        userPoolClientId: process.env.NEXT_PUBLIC_COGNITO_CLIENT_ID!,
        signUpVerificationMethod: 'code',
        loginWith: {
          email: true,
          username: false,
          phone: false,
        },
```

```
        userAttributes: {
          email: { required: true },
        },
        mfa: {
          status: 'optional',
          totpEnabled: true,
          smsEnabled: false,
        },
        passwordFormat: {
          minLength: 12,
          requireLowercase: true,
          requireUppercase: true,
          requireNumbers: true,
          requireSpecialCharacters: true,
        },
      },
    },
  }, {
    ssr: true,
  });
}
```

lib/auth/context.tsx

```
'use client';

import {
  createContext,
  useContext,
  useEffect,
  useState,
  useCallback,
  type ReactNode,
} from 'react';
import {
  getCurrentUser,
  fetchAuthSession,
  signIn,
  signOut,
  confirmSignIn,
  type AuthUser,
} from 'aws-amplify/auth';

// ============================================================================
// TYPES
// ============================================================================
```

```tsx
export type AdminRole = 'super_admin' | 'admin' | 'operator' | 'auditor';

export interface AdminUser {
  id: string;
  email: string;
  firstName: string;
  lastName: string;
  displayName: string;
  role: AdminRole;
  permissions: string[];
  mfaEnabled: boolean;
  lastLoginAt?: string;
}

interface AuthState {
  isAuthenticated: boolean;
  isLoading: boolean;
  user: AdminUser | null;
  accessToken: string | null;
  error: string | null;
}

interface AuthContextValue extends AuthState {
  login: (email: string, password: string) => Promise<{ needsMfa: boolean }>;
  confirmMfa: (code: string) => Promise<void>;
  logout: () => Promise<void>;
  refreshSession: () => Promise<void>;
  hasPermission: (permission: string) => boolean;
}

// ============================================================================
// CONTEXT
// ============================================================================

const AuthContext = createContext<AuthContextValue | null>(null);

export function AuthProvider({ children }: { children: ReactNode }) {
  const [state, setState] = useState<AuthState>({
    isAuthenticated: false,
    isLoading: true,
    user: null,
    accessToken: null,
    error: null,
  });
```

```
// Check session on mount
useEffect(() => {
  checkSession();
}, []);

const checkSession = async () => {
  try {
    const user = await getCurrentUser();
    const session = await fetchAuthSession();

    if (session.tokens?.accessToken) {
      // Fetch admin profile from API
      const adminProfile = await fetchAdminProfile(
        session.tokens.accessToken.toString()
      );

      setState({
        isAuthenticated: true,
        isLoading: false,
        user: adminProfile,
        accessToken: session.tokens.accessToken.toString(),
        error: null,
      });
    } else {
      throw new Error('No access token');
    }
  } catch {
    setState({
      isAuthenticated: false,
      isLoading: false,
      user: null,
      accessToken: null,
      error: null,
    });
  }
};

const fetchAdminProfile = async (token: string): Promise<AdminUser> => {
  const response = await fetch(
    `${process.env.NEXT_PUBLIC_API_URL}/api/v2/admin/profile`,
    {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    }
  );
```

```
  if (!response.ok) {
    throw new Error('Failed to fetch admin profile');
  }

  return response.json();
};

const login = useCallback(async (email: string, password: string) => {
  try {
    setState(prev => ({ ...prev, isLoading: true, error: null }));

    const result = await signIn({
      username: email,
      password,
    });

    if (result.nextStep?.signInStep === 'CONFIRM_SIGN_IN_WITH_TOTP_CODE') {
      setState(prev => ({ ...prev, isLoading: false }));
      return { needsMfa: true };
    }

    if (result.isSignedIn) {
      await checkSession();
    }

    return { needsMfa: false };
  } catch (error) {
    setState(prev => ({
      ...prev,
      isLoading: false,
      error: error instanceof Error ? error.message : 'Login failed',
    }));
    throw error;
  }
}, []);

const confirmMfa = useCallback(async (code: string) => {
  try {
    setState(prev => ({ ...prev, isLoading: true, error: null }));

    const result = await confirmSignIn({
      challengeResponse: code,
    });

    if (result.isSignedIn) {
```

```
      await checkSession();
    }
  } catch (error) {
    setState(prev => ({
      ...prev,
      isLoading: false,
      error: error instanceof Error ? error.message : 'MFA verification failed',
    }));
    throw error;
  }
}, []);

const logout = useCallback(async () => {
  try {
    await signOut();
    setState({
      isAuthenticated: false,
      isLoading: false,
      user: null,
      accessToken: null,
      error: null,
    });
  } catch (error) {
    console.error('Logout failed:', error);
  }
}, []);

const refreshSession = useCallback(async () => {
  await checkSession();
}, []);

const hasPermission = useCallback((permission: string): boolean => {
  if (!state.user) return false;

  // Super admin has all permissions
  if (state.user.role === 'super_admin') return true;

  // Check for wildcard permissions
  const parts = permission.split(':');
  if (parts.length === 2) {
    const wildcardPermission = `${parts[0]}:*`;
    if (state.user.permissions.includes(wildcardPermission)) return true;
  }

  // Check exact permission
  return state.user.permissions.includes(permission);
```

```
  }, [state.user]);

  return (
    <AuthContext.Provider
      value={{
        ...state,
        login,
        confirmMfa,
        logout,
        refreshSession,
        hasPermission,
      }}
    >
      {children}
    </AuthContext.Provider>
  );
}

export function useAuth() {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error('useAuth must be used within an AuthProvider');
  }
  return context;
}

export function useCurrentAdmin() {
  const { user, isAuthenticated } = useAuth();

  if (!isAuthenticated || !user) {
    throw new Error('Not authenticated');
  }

  return user;
}
```

lib/auth/hooks.ts

```
'use client';

import { useAuth } from './context';
import { useRouter, usePathname } from 'next/navigation';
import { useEffect } from 'react';

/**
 * Hook to protect routes that require authentication
```

```
 */
export function useRequireAuth(requiredPermission?: string) {
  const { isAuthenticated, isLoading, hasPermission, user } = useAuth();
  const router = useRouter();
  const pathname = usePathname();

  useEffect(() => {
    if (isLoading) return;

    if (!isAuthenticated) {
      const returnUrl = encodeURIComponent(pathname);
      router.push(`/login?returnUrl=${returnUrl}`);
      return;
    }

    if (requiredPermission && !hasPermission(requiredPermission)) {
      router.push('/unauthorized');
    }
  }, [isAuthenticated, isLoading, hasPermission, requiredPermission, pathname, router]);

  return {
    isLoading,
    isAuthenticated,
    user,
    hasPermission,
  };
}

/**
 * Hook to check if user can perform production actions
 */
export function useProductionAccess() {
  const { user, hasPermission } = useAuth();

  const canAccessProduction = user?.mfaEnabled && hasPermission('deployments:prod');

  return {
    canAccessProduction,
    mfaRequired: !user?.mfaEnabled,
    hasPermission: hasPermission('deployments:prod'),
  };
}
```

PART 3: API CLIENT

lib/api/client.ts

```typescript
/**
 * Type-safe API client for RADIANT Admin Dashboard
 * Handles authentication, error handling, and request/response transformation
 */

import { fetchAuthSession } from 'aws-amplify/auth';

// ============================================================================
// TYPES
// ============================================================================

export interface ApiError {
  code: string;
  message: string;
  details?: Record<string, unknown>;
  requestId?: string;
}

export interface ApiResponse<T> {
  data: T;
  meta?: {
    page?: number;
    pageSize?: number;
    total?: number;
    hasMore?: boolean;
  };
}

export interface PaginationParams {
  page?: number;
  pageSize?: number;
  sortBy?: string;
  sortOrder?: 'asc' | 'desc';
}

type HttpMethod = 'GET' | 'POST' | 'PUT' | 'PATCH' | 'DELETE';

interface RequestOptions {
  method?: HttpMethod;
  body?: unknown;
  params?: Record<string, string | number | boolean | undefined>;
  headers?: Record<string, string>;
```

```
  skipAuth?: boolean;
}

// ============================================================================
// API CLIENT
// ============================================================================

const API_BASE_URL = process.env.NEXT_PUBLIC_API_URL || '';

class ApiClient {
  private baseUrl: string;

  constructor(baseUrl: string) {
    this.baseUrl = baseUrl;
  }

  private async getAuthToken(): Promise<string | null> {
    try {
      const session = await fetchAuthSession();
      return session.tokens?.accessToken?.toString() || null;
    } catch {
      return null;
    }
  }

  private buildUrl(
    path: string,
    params?: Record<string, string | number | boolean | undefined>
  ): string {
    const url = new URL(path, this.baseUrl);

    if (params) {
      Object.entries(params).forEach(([key, value]) => {
        if (value !== undefined) {
          url.searchParams.append(key, String(value));
        }
      });
    }

    return url.toString();
  }

  async request<T>(path: string, options: RequestOptions = {}): Promise<T> {
    const {
      method = 'GET',
      body,
```

```
  params,
  headers = {},
  skipAuth = false,
} = options;

const url = this.buildUrl(path, params);

const requestHeaders: Record<string, string> = {
  'Content-Type': 'application/json',
  ...headers,
};

if (!skipAuth) {
  const token = await this.getAuthToken();
  if (token) {
    requestHeaders['Authorization'] = `Bearer ${token}`;
  }
}

const requestInit: RequestInit = {
  method,
  headers: requestHeaders,
  credentials: 'include',
};

if (body && method !== 'GET') {
  requestInit.body = JSON.stringify(body);
}

const response = await fetch(url, requestInit);

// Handle non-JSON responses
const contentType = response.headers.get('content-type');
if (!contentType?.includes('application/json')) {
  if (!response.ok) {
    throw {
      code: 'NETWORK_ERROR',
      message: `Request failed with status ${response.status}`,
      requestId: response.headers.get('x-request-id') || undefined,
    } as ApiError;
  }
  return {} as T;
}

const data = await response.json();
```

```
    if (!response.ok) {
      throw {
        code: data.error?.code || 'UNKNOWN_ERROR',
        message: data.error?.message || 'An unknown error occurred',
        details: data.error?.details,
        requestId: response.headers.get('x-request-id') || data.requestId,
      } as ApiError;
    }

    return data;
  }

  // Convenience methods
  get<T>(path: string, params?: Record<string, string | number | boolean | undefined>): Pro
    return this.request<T>(path, { method: 'GET', params });
  }

  post<T>(path: string, body?: unknown): Promise<T> {
    return this.request<T>(path, { method: 'POST', body });
  }

  put<T>(path: string, body?: unknown): Promise<T> {
    return this.request<T>(path, { method: 'PUT', body });
  }

  patch<T>(path: string, body?: unknown): Promise<T> {
    return this.request<T>(path, { method: 'PATCH', body });
  }

  delete<T>(path: string): Promise<T> {
    return this.request<T>(path, { method: 'DELETE' });
  }
}

// Export singleton instance
export const api = new ApiClient(API_BASE_URL);
```

lib/api/types.ts

```
/**
 * API Response Types for RADIANT Admin Dashboard
 */

// =============================================================================
// MODELS
// =============================================================================
```

```typescript
export type ThermalState = 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';
export type ServiceState = 'RUNNING' | 'DEGRADED' | 'DISABLED' | 'OFFLINE';
export type ModelCategory =
  | 'vision_classification'
  | 'vision_detection'
  | 'vision_segmentation'
  | 'audio_stt'
  | 'audio_speaker'
  | 'scientific_protein'
  | 'scientific_math'
  | 'medical_imaging'
  | 'geospatial'
  | 'generative_3d'
  | 'llm_text';

export interface Model {
  id: string;
  name: string;
  displayName: string;
  description: string;
  category: ModelCategory;
  specialty: string;
  provider: string;
  isExternal: boolean;
  isEnabled: boolean;
  thermalState: ThermalState;
  serviceState: ServiceState;
  parameters: number;
  accuracy?: string;
  capabilities: string[];
  inputFormats: string[];
  outputFormats: string[];
  minTier: number;
  pricing: {
    inputPer1k: number;
    outputPer1k: number;
    hourlyRate?: number;
    perImage?: number;
    perMinuteAudio?: number;
    markup: number;
  };
  usage: {
    last24h: number;
    last7d: number;
    last30d: number;
```

```
  };
  status: 'active' | 'beta' | 'deprecated' | 'coming_soon';
  createdAt: string;
  updatedAt: string;
}

export interface ModelFilters {
  search?: string;
  category?: ModelCategory;
  thermalState?: ThermalState;
  isExternal?: boolean;
  isEnabled?: boolean;
  minTier?: number;
}

// ============================================================================
// PROVIDERS
// ============================================================================

export interface Provider {
  id: string;
  name: string;
  displayName: string;
  description: string;
  category: string;
  isEnabled: boolean;
  hasApiKey: boolean;
  apiKeyMasked?: string;
  baseUrl?: string;
  modelCount: number;
  healthStatus: 'healthy' | 'degraded' | 'down' | 'unknown';
  lastHealthCheck?: string;
  rateLimits: {
    requestsPerMinute: number;
    tokensPerMinute: number;
  };
  createdAt: string;
  updatedAt: string;
}

// ============================================================================
// SERVICES
// ============================================================================

export interface MidLevelService {
  id: string;
```

```
  name: string;
  displayName: string;
  description: string;
  state: ServiceState;
  models: string[];
  healthStatus: 'healthy' | 'degraded' | 'down';
  lastHealthCheck: string;
  metrics: {
    requestsLast24h: number;
    avgLatencyMs: number;
    errorRate: number;
  };
}

// ============================================================================
// ADMINISTRATORS
// ============================================================================

export type AdminRole = 'super_admin' | 'admin' | 'operator' | 'auditor';

export interface Administrator {
  id: string;
  email: string;
  firstName: string;
  lastName: string;
  displayName: string;
  role: AdminRole;
  mfaEnabled: boolean;
  status: 'active' | 'pending' | 'inactive' | 'suspended';
  lastLoginAt?: string;
  createdAt: string;
}

export interface Invitation {
  id: string;
  email: string;
  role: AdminRole;
  invitedBy: string;
  invitedByName: string;
  message?: string;
  status: 'pending' | 'accepted' | 'expired' | 'revoked';
  expiresAt: string;
  createdAt: string;
}

export interface ApprovalRequest {
```

```
  id: string;
  type: 'deployment' | 'promotion' | 'model_activation' | 'provider_change' | 'user_role_cha
  title: string;
  description: string;
  environment: 'dev' | 'staging' | 'prod';
  riskLevel: 'low' | 'medium' | 'high' | 'critical';
  status: 'pending' | 'approved' | 'rejected' | 'expired' | 'executed';
  initiatedBy: {
    id: string;
    name: string;
    email: string;
  };
  approvedBy?: {
    id: string;
    name: string;
    email: string;
  };
  initiatedAt: string;
  expiresAt: string;
  approvedAt?: string;
  payload: Record<string, unknown>;
}

// ============================================================================
// BILLING
// ============================================================================

export interface BillingSummary {
  currentMonth: {
    totalCost: number;
    totalRevenue: number;
    margin: number;
    breakdown: {
      external: number;
      selfHosted: number;
      infrastructure: number;
    };
  };
  comparison: {
    previousMonth: number;
    percentChange: number;
  };
  projections: {
    endOfMonth: number;
    endOfQuarter: number;
  };
```

```
}

export interface UsageStats {
  period: 'hourly' | 'daily' | 'weekly' | 'monthly';
  data: Array<{
    timestamp: string;
    requests: number;
    tokens: {
      input: number;
      output: number;
    };
    cost: number;
    revenue: number;
  }>;
}

export interface MarginConfig {
  providerId: string;
  providerName: string;
  defaultMargin: number;
  modelOverrides: Array<{
    modelId: string;
    modelName: string;
    margin: number;
  }>;
}

// ============================================================================
// GEOGRAPHIC
// ============================================================================

export interface RegionStats {
  region: string;
  displayName: string;
  status: 'active' | 'standby' | 'maintenance';
  requests: {
    last24h: number;
    last7d: number;
  };
  latency: {
    avg: number;
    p95: number;
    p99: number;
  };
  endpoints: {
    total: number;
```

```
    healthy: number;
  };
}

// ============================================================================
// DASHBOARD
// ============================================================================

export interface DashboardMetrics {
  totalRequests: {
    value: number;
    change: number;
    period: '24h' | '7d' | '30d';
  };
  activeModels: {
    value: number;
    external: number;
    selfHosted: number;
  };
  revenue: {
    value: number;
    change: number;
    period: '24h' | '7d' | '30d';
  };
  errorRate: {
    value: number;
    change: number;
    period: '24h' | '7d' | '30d';
  };
}

export interface SystemHealth {
  overall: 'healthy' | 'degraded' | 'critical';
  components: Array<{
    name: string;
    status: 'healthy' | 'degraded' | 'down';
    lastCheck: string;
    message?: string;
  }>;
}

export interface RecentActivity {
  id: string;
  type: 'model_activation' | 'provider_update' | 'admin_action' | 'deployment' | 'alert';
  title: string;
  description: string;
```

```
  actor?: {
    id: string;
    name: string;
  };
  timestamp: string;
  metadata?: Record<string, unknown>;
}

// ============================================================================
// NOTIFICATIONS
// ============================================================================

export interface Notification {
  id: string;
  type: 'info' | 'warning' | 'error' | 'success';
  title: string;
  message: string;
  isRead: boolean;
  actionUrl?: string;
  createdAt: string;
}

export interface NotificationPreferences {
  email: {
    enabled: boolean;
    digestFrequency: 'immediate' | 'hourly' | 'daily' | 'weekly';
    types: {
      approvalRequests: boolean;
      modelAlerts: boolean;
      billingAlerts: boolean;
      systemHealth: boolean;
    };
  };
  slack?: {
    enabled: boolean;
    webhookConfigured: boolean;
    types: {
      approvalRequests: boolean;
      modelAlerts: boolean;
      billingAlerts: boolean;
      systemHealth: boolean;
    };
  };
  inApp: {
    enabled: boolean;
    playSound: boolean;
```

```
  };
}
```

lib/api/endpoints.ts

```
/**
 * API Endpoint Definitions
 */

import { api } from './client';
import type {
  Model,
  ModelFilters,
  Provider,
  MidLevelService,
  Administrator,
  Invitation,
  ApprovalRequest,
  BillingSummary,
  UsageStats,
  MarginConfig,
  RegionStats,
  DashboardMetrics,
  SystemHealth,
  RecentActivity,
  Notification,
  NotificationPreferences,
  ThermalState,
  ServiceState,
  AdminRole,
} from './types';
import type { PaginationParams, ApiResponse } from './client';

// ============================================================================
// DASHBOARD
// ============================================================================

export const dashboardApi = {
  getMetrics: (period: '24h' | '7d' | '30d' = '24h') =>
    api.get<DashboardMetrics>('/api/v2/admin/dashboard/metrics', { period }),

  getHealth: () =>
    api.get<SystemHealth>('/api/v2/admin/dashboard/health'),

  getRecentActivity: (limit = 10) =>
    api.get<RecentActivity[]>('/api/v2/admin/dashboard/activity', { limit }),
```

```
};

// ============================================================================
// MODELS
// ============================================================================

export const modelsApi = {
  list: (filters?: ModelFilters & PaginationParams) =>
    api.get<ApiResponse<Model[]>>('/api/v2/admin/models', filters as Record<string, string

  get: (id: string) =>
    api.get<Model>(`/api/v2/admin/models/${id}`),

  update: (id: string, data: Partial<Model>) =>
    api.patch<Model>(`/api/v2/admin/models/${id}`, data),

  setThermalState: (id: string, state: ThermalState) =>
    api.post<Model>(`/api/v2/admin/models/${id}/thermal`, { state }),

  setEnabled: (id: string, enabled: boolean) =>
    api.post<Model>(`/api/v2/admin/models/${id}/enabled`, { enabled }),

  warmUp: (id: string) =>
    api.post<{ estimatedWaitSeconds: number }>(`/api/v2/admin/models/${id}/warm-up`),

  getUsage: (id: string, period: '24h' | '7d' | '30d') =>
    api.get<UsageStats>(`/api/v2/admin/models/${id}/usage`, { period }),
};

// ============================================================================
// PROVIDERS
// ============================================================================

export const providersApi = {
  list: (params?: PaginationParams) =>
    api.get<ApiResponse<Provider[]>>('/api/v2/admin/providers', params as Record<string, str

  get: (id: string) =>
    api.get<Provider>(`/api/v2/admin/providers/${id}`),

  update: (id: string, data: Partial<Provider>) =>
    api.patch<Provider>(`/api/v2/admin/providers/${id}`, data),

  setApiKey: (id: string, apiKey: string) =>
    api.post<{ success: boolean }>(`/api/v2/admin/providers/${id}/api-key`, { apiKey }),
```

```
  testConnection: (id: string) =>
    api.post<{ healthy: boolean; latencyMs: number; error?: string }>(`/api/v2/admin/provide

  setEnabled: (id: string, enabled: boolean) =>
    api.post<Provider>(`/api/v2/admin/providers/${id}/enabled`, { enabled }),
};

// ============================================================================
// SERVICES
// ============================================================================

export const servicesApi = {
  list: () =>
    api.get<MidLevelService[]>('/api/v2/admin/services'),

  get: (id: string) =>
    api.get<MidLevelService>(`/api/v2/admin/services/${id}`),

  setState: (id: string, state: ServiceState) =>
    api.post<MidLevelService>(`/api/v2/admin/services/${id}/state`, { state }),

  getHealth: (id: string) =>
    api.get<{ healthy: boolean; checks: Array<{ name: string; passed: boolean }> }>(
      `/api/v2/admin/services/${id}/health`
    ),
};

// ============================================================================
// ADMINISTRATORS
// ============================================================================

export const administratorsApi = {
  list: (params?: PaginationParams) =>
    api.get<ApiResponse<Administrator[]>>('/api/v2/admin/administrators', params as Record<s

  get: (id: string) =>
    api.get<Administrator>(`/api/v2/admin/administrators/${id}`),

  update: (id: string, data: Partial<Administrator>) =>
    api.patch<Administrator>(`/api/v2/admin/administrators/${id}`, data),

  deactivate: (id: string) =>
    api.post<{ success: boolean }>(`/api/v2/admin/administrators/${id}/deactivate`),

  changeRole: (id: string, role: AdminRole) =>
    api.post<Administrator>(`/api/v2/admin/administrators/${id}/role`, { role }),
```

```
};

// ============================================================================
// INVITATIONS
// ============================================================================

export const invitationsApi = {
  list: (status?: 'pending' | 'accepted' | 'expired' | 'revoked') =>
    api.get<Invitation[]>('/api/v2/admin/invitations', { status }),

  create: (data: { email: string; role: AdminRole; message?: string }) =>
    api.post<Invitation>('/api/v2/admin/invitations', data),

  revoke: (id: string) =>
    api.post<{ success: boolean }>(`/api/v2/admin/invitations/${id}/revoke`),

  resend: (id: string) =>
    api.post<{ success: boolean }>(`/api/v2/admin/invitations/${id}/resend`),

  accept: (token: string, data: { firstName: string; lastName: string; password: string }) =
    api.post<{ success: boolean }>('/api/v2/admin/invitations/accept', { token, ...data }),
};

// ============================================================================
// APPROVALS
// ============================================================================

export const approvalsApi = {
  list: (status?: 'pending' | 'approved' | 'rejected') =>
    api.get<ApprovalRequest[]>('/api/v2/admin/approvals', { status }),

  get: (id: string) =>
    api.get<ApprovalRequest>(`/api/v2/admin/approvals/${id}`),

  approve: (id: string, notes?: string) =>
    api.post<ApprovalRequest>(`/api/v2/admin/approvals/${id}/approve`, { notes }),

  reject: (id: string, reason: string) =>
    api.post<ApprovalRequest>(`/api/v2/admin/approvals/${id}/reject`, { reason }),
};

// ============================================================================
// BILLING
// ============================================================================

export const billingApi = {
```

```
  getSummary: () =>
    api.get<BillingSummary>('/api/v2/admin/billing/summary'),

  getUsage: (period: 'hourly' | 'daily' | 'weekly' | 'monthly') =>
    api.get<UsageStats>('/api/v2/admin/billing/usage', { period }),

  getMargins: () =>
    api.get<MarginConfig[]>('/api/v2/admin/billing/margins'),

  updateMargins: (providerId: string, margins: Partial<MarginConfig>) =>
    api.patch<MarginConfig>(`/api/v2/admin/billing/margins/${providerId}`, margins),
};

// ============================================================================
// GEOGRAPHIC
// ============================================================================

export const geographicApi = {
  getRegions: () =>
    api.get<RegionStats[]>('/api/v2/admin/geographic/regions'),

  getRegion: (region: string) =>
    api.get<RegionStats>(`/api/v2/admin/geographic/regions/${region}`),
};

// ============================================================================
// NOTIFICATIONS
// ============================================================================

export const notificationsApi = {
  list: (unreadOnly = false) =>
    api.get<Notification[]>('/api/v2/admin/notifications', { unreadOnly }),

  markRead: (id: string) =>
    api.post<{ success: boolean }>(`/api/v2/admin/notifications/${id}/read`),

  markAllRead: () =>
    api.post<{ success: boolean }>('/api/v2/admin/notifications/read-all'),

  getPreferences: () =>
    api.get<NotificationPreferences>('/api/v2/admin/notifications/preferences'),

  updatePreferences: (prefs: Partial<NotificationPreferences>) =>
    api.patch<NotificationPreferences>('/api/v2/admin/notifications/preferences', prefs),
};
```

```typescript
// ============================================================================
// PROFILE
// ============================================================================

export const profileApi = {
  get: () =>
    api.get<Administrator>('/api/v2/admin/profile'),

  update: (data: { firstName?: string; lastName?: string; displayName?: string }) =>
    api.patch<Administrator>('/api/v2/admin/profile', data),

  enableMfa: () =>
    api.post<{ qrCodeUrl: string; secret: string }>('/api/v2/admin/profile/mfa/enable'),

  verifyMfa: (code: string) =>
    api.post<{ success: boolean }>('/api/v2/admin/profile/mfa/verify', { code }),

  disableMfa: (code: string) =>
    api.post<{ success: boolean }>('/api/v2/admin/profile/mfa/disable', { code }),
};
```

PART 4: REACT QUERY HOOKS

lib/hooks/use-models.ts

```typescript
'use client';

import {
  useQuery,
  useMutation,
  useQueryClient,
  type UseQueryOptions,
} from '@tanstack/react-query';
import { modelsApi } from '@/lib/api/endpoints';
import type { Model, ModelFilters, ThermalState } from '@/lib/api/types';
import type { PaginationParams, ApiResponse } from '@/lib/api/client';

// Query keys
export const modelKeys = {
  all: ['models'] as const,
  lists: () => [...modelKeys.all, 'list'] as const,
  list: (filters?: ModelFilters & PaginationParams) =>
    [...modelKeys.lists(), filters] as const,
  details: () => [...modelKeys.all, 'detail'] as const,
  detail: (id: string) => [...modelKeys.details(), id] as const,
```

```typescript
  usage: (id: string, period: string) =>
    [...modelKeys.detail(id), 'usage', period] as const,
};

/**
 * Hook to fetch models list with filtering and pagination
 */
export function useModels(
  filters?: ModelFilters & PaginationParams,
  options?: Omit<UseQueryOptions<ApiResponse<Model[]>>, 'queryKey' | 'queryFn'>
) {
  return useQuery({
    queryKey: modelKeys.list(filters),
    queryFn: () => modelsApi.list(filters),
    ...options,
  });
}

/**
 * Hook to fetch a single model
 */
export function useModel(
  id: string,
  options?: Omit<UseQueryOptions<Model>, 'queryKey' | 'queryFn'>
) {
  return useQuery({
    queryKey: modelKeys.detail(id),
    queryFn: () => modelsApi.get(id),
    enabled: !!id,
    ...options,
  });
}

/**
 * Hook to update model thermal state
 */
export function useSetThermalState() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: ({ id, state }: { id: string; state: ThermalState }) =>
      modelsApi.setThermalState(id, state),
    onSuccess: (updatedModel) => {
      // Update the specific model in cache
      queryClient.setQueryData(modelKeys.detail(updatedModel.id), updatedModel);
      // Invalidate lists to refresh
```

```
      queryClient.invalidateQueries({ queryKey: modelKeys.lists() });
    },
  });
}

/**
 * Hook to enable/disable a model
 */
export function useSetModelEnabled() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: ({ id, enabled }: { id: string; enabled: boolean }) =>
      modelsApi.setEnabled(id, enabled),
    onSuccess: (updatedModel) => {
      queryClient.setQueryData(modelKeys.detail(updatedModel.id), updatedModel);
      queryClient.invalidateQueries({ queryKey: modelKeys.lists() });
    },
  });
}

/**
 * Hook to warm up a model
 */
export function useWarmUpModel() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: (id: string) => modelsApi.warmUp(id),
    onSuccess: (_, id) => {
      // Invalidate to refresh state
      queryClient.invalidateQueries({ queryKey: modelKeys.detail(id) });
    },
  });
}

/**
 * Hook to fetch model usage statistics
 */
export function useModelUsage(
  id: string,
  period: '24h' | '7d' | '30d' = '24h'
) {
  return useQuery({
    queryKey: modelKeys.usage(id, period),
    queryFn: () => modelsApi.getUsage(id, period),
```

```
    enabled: !!id,
  });
}
```

lib/hooks/use-administrators.ts

```
'use client';

import {
  useQuery,
  useMutation,
  useQueryClient,
} from '@tanstack/react-query';
import {
  administratorsApi,
  invitationsApi,
  approvalsApi,
} from '@/lib/api/endpoints';
import type {
  Administrator,
  Invitation,
  ApprovalRequest,
  AdminRole,
} from '@/lib/api/types';
import type { PaginationParams, ApiResponse } from '@/lib/api/client';

// Query keys
export const adminKeys = {
  all: ['administrators'] as const,
  lists: () => [...adminKeys.all, 'list'] as const,
  list: (params?: PaginationParams) => [...adminKeys.lists(), params] as const,
  details: () => [...adminKeys.all, 'detail'] as const,
  detail: (id: string) => [...adminKeys.details(), id] as const,
};

export const invitationKeys = {
  all: ['invitations'] as const,
  list: (status?: string) => [...invitationKeys.all, status] as const,
};

export const approvalKeys = {
  all: ['approvals'] as const,
  list: (status?: string) => [...approvalKeys.all, status] as const,
  detail: (id: string) => [...approvalKeys.all, 'detail', id] as const,
};
```

```typescript
/**
 * Hook to fetch administrators list
 */
export function useAdministrators(params?: PaginationParams) {
  return useQuery({
    queryKey: adminKeys.list(params),
    queryFn: () => administratorsApi.list(params),
  });
}

/**
 * Hook to fetch a single administrator
 */
export function useAdministrator(id: string) {
  return useQuery({
    queryKey: adminKeys.detail(id),
    queryFn: () => administratorsApi.get(id),
    enabled: !!id,
  });
}

/**
 * Hook to change administrator role
 */
export function useChangeAdminRole() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: ({ id, role }: { id: string; role: AdminRole }) =>
      administratorsApi.changeRole(id, role),
    onSuccess: (updatedAdmin) => {
      queryClient.setQueryData(adminKeys.detail(updatedAdmin.id), updatedAdmin);
      queryClient.invalidateQueries({ queryKey: adminKeys.lists() });
    },
  });
}

/**
 * Hook to deactivate an administrator
 */
export function useDeactivateAdmin() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: (id: string) => administratorsApi.deactivate(id),
    onSuccess: (_, id) => {
```

```
      queryClient.invalidateQueries({ queryKey: adminKeys.detail(id) });
      queryClient.invalidateQueries({ queryKey: adminKeys.lists() });
    },
  });
}

// ============================================================================
// INVITATIONS
// ============================================================================

/**
 * Hook to fetch invitations
 */
export function useInvitations(status?: 'pending' | 'accepted' | 'expired' | 'revoked') {
  return useQuery({
    queryKey: invitationKeys.list(status),
    queryFn: () => invitationsApi.list(status),
  });
}

/**
 * Hook to create an invitation
 */
export function useCreateInvitation() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: (data: { email: string; role: AdminRole; message?: string }) =>
      invitationsApi.create(data),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: invitationKeys.all });
    },
  });
}

/**
 * Hook to revoke an invitation
 */
export function useRevokeInvitation() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: (id: string) => invitationsApi.revoke(id),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: invitationKeys.all });
    },
```

```
  });
}

/**
 * Hook to resend an invitation
 */
export function useResendInvitation() {
  return useMutation({
    mutationFn: (id: string) => invitationsApi.resend(id),
  });
}

// ============================================================================
// APPROVALS
// ============================================================================

/**
 * Hook to fetch approval requests
 */
export function useApprovals(status?: 'pending' | 'approved' | 'rejected') {
  return useQuery({
    queryKey: approvalKeys.list(status),
    queryFn: () => approvalsApi.list(status),
    refetchInterval: 30000, // Refresh every 30 seconds
  });
}

/**
 * Hook to fetch a single approval request
 */
export function useApproval(id: string) {
  return useQuery({
    queryKey: approvalKeys.detail(id),
    queryFn: () => approvalsApi.get(id),
    enabled: !!id,
  });
}

/**
 * Hook to approve a request
 */
export function useApproveRequest() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: ({ id, notes }: { id: string; notes?: string }) =>
```

```
      approvalsApi.approve(id, notes),
    onSuccess: (_, { id }) => {
      queryClient.invalidateQueries({ queryKey: approvalKeys.detail(id) });
      queryClient.invalidateQueries({ queryKey: approvalKeys.all });
    },
  });
}

/**
 * Hook to reject a request
 */
export function useRejectRequest() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: ({ id, reason }: { id: string; reason: string }) =>
      approvalsApi.reject(id, reason),
    onSuccess: (_, { id }) => {
      queryClient.invalidateQueries({ queryKey: approvalKeys.detail(id) });
      queryClient.invalidateQueries({ queryKey: approvalKeys.all });
    },
  });
}
```

lib/hooks/use-billing.ts

```
'use client';

import {
  useQuery,
  useMutation,
  useQueryClient,
} from '@tanstack/react-query';
import { billingApi } from '@/lib/api/endpoints';
import type { BillingSummary, UsageStats, MarginConfig } from '@/lib/api/types';

// Query keys
export const billingKeys = {
  all: ['billing'] as const,
  summary: () => [...billingKeys.all, 'summary'] as const,
  usage: (period: string) => [...billingKeys.all, 'usage', period] as const,
  margins: () => [...billingKeys.all, 'margins'] as const,
};

/**
 * Hook to fetch billing summary
```

```tsx
 */
export function useBillingSummary() {
  return useQuery({
    queryKey: billingKeys.summary(),
    queryFn: () => billingApi.getSummary(),
    refetchInterval: 60000, // Refresh every minute
  });
}

/**
 * Hook to fetch usage statistics
 */
export function useUsageStats(period: 'hourly' | 'daily' | 'weekly' | 'monthly' = 'daily') {
  return useQuery({
    queryKey: billingKeys.usage(period),
    queryFn: () => billingApi.getUsage(period),
  });
}

/**
 * Hook to fetch margins configuration
 */
export function useMargins() {
  return useQuery({
    queryKey: billingKeys.margins(),
    queryFn: () => billingApi.getMargins(),
  });
}

/**
 * Hook to update margins
 */
export function useUpdateMargins() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: ({ providerId, margins }: { providerId: string; margins: Partial<MarginConfi
      billingApi.updateMargins(providerId, margins),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: billingKeys.margins() });
    },
  });
}
```

PART 5: LAYOUT COMPONENTS

components/layout/sidebar.tsx

```
'use client';

import Link from 'next/link';
import { usePathname } from 'next/navigation';
import {
  LayoutDashboard,
  Globe,
  Cpu,
  Layers,
  Cloud,
  Users,
  CreditCard,
  BarChart3,
  Bell,
  Settings,
  LogOut,
  ChevronRight,
  Shield,
  ClipboardCheck,
} from 'lucide-react';
import { cn } from '@/lib/utils/cn';
import { useAuth } from '@/lib/auth/context';

interface NavItem {
  title: string;
  href: string;
  icon: React.ComponentType<{ className?: string }>;
  permission?: string;
  children?: NavItem[];
  badge?: number | string;
}

const navItems: NavItem[] = [
  {
    title: 'Dashboard',
    href: '/',
    icon: LayoutDashboard,
  },
  {
    title: 'Geographic',
    href: '/geographic',
    icon: Globe,
```

```
},
{
  title: 'AI Models',
  href: '/models',
  icon: Cpu,
},
{
  title: 'Services',
  href: '/services',
  icon: Layers,
},
{
  title: 'Providers',
  href: '/providers',
  icon: Cloud,
},
{
  title: 'Administrators',
  href: '/administrators',
  icon: Users,
  permission: 'admin:read',
  children: [
    { title: 'All Admins', href: '/administrators', icon: Users },
    { title: 'Invitations', href: '/administrators/invitations', icon: Shield },
    { title: 'Approvals', href: '/administrators/approvals', icon: ClipboardCheck },
  ],
},
{
  title: 'Billing',
  href: '/billing',
  icon: CreditCard,
  permission: 'billing:read',
  children: [
    { title: 'Overview', href: '/billing', icon: CreditCard },
    { title: 'Margins', href: '/billing/margins', icon: BarChart3 },
    { title: 'Invoices', href: '/billing/invoices', icon: CreditCard },
  ],
},
{
  title: 'Usage',
  href: '/usage',
  icon: BarChart3,
},
{
  title: 'Notifications',
  href: '/notifications',
```

```
    icon: Bell,
  },
  {
    title: 'Settings',
    href: '/settings',
    icon: Settings,
    children: [
      { title: 'General', href: '/settings', icon: Settings },
      { title: 'Profile', href: '/settings/profile', icon: Users },
      { title: 'Security', href: '/settings/security', icon: Shield },
    ],
  },
];

export function Sidebar() {
  const pathname = usePathname();
  const { user, logout, hasPermission } = useAuth();

  const filteredItems = navItems.filter(
    (item) => !item.permission || hasPermission(item.permission)
  );

  return (
    <aside className="flex h-screen w-64 flex-col border-r border-sidebar-border bg-sidebar'
      {/* Logo */}
      <div className="flex h-16 items-center gap-2 border-b border-sidebar-border px-4">
        <div className="flex h-8 w-8 items-center justify-center rounded-lg bg-primary">
          <span className="text-lg font-bold text-primary-foreground">R</span>
        </div>
        <div>
          <h1 className="text-lg font-semibold text-sidebar-foreground">RADIANT</h1>
          <p className="text-xs text-muted-foreground">Admin Dashboard</p>
        </div>
      </div>

      {/* Navigation */}
      <nav className="flex-1 overflow-y-auto p-4">
        <ul className="space-y-1">
          {filteredItems.map((item) => (
            <NavItemComponent
              key={item.href}
              item={item}
              pathname={pathname}
            />
          ))}
        </ul>
```

```
      </nav>

      {/* User Profile & Logout */}
      <div className="border-t border-sidebar-border p-4">
        <div className="flex items-center gap-3 rounded-lg px-3 py-2">
          <div className="flex h-8 w-8 items-center justify-center rounded-full bg-primary/1
            <span className="text-sm font-medium text-primary">
              {user?.firstName?.[0]}{user?.lastName?.[0]}
            </span>
          </div>
          <div className="flex-1 overflow-hidden">
            <p className="truncate text-sm font-medium text-sidebar-foreground">
              {user?.displayName || `${user?.firstName} ${user?.lastName}`}
            </p>
            <p className="truncate text-xs text-muted-foreground capitalize">
              {user?.role?.replace('_', ' ')}
            </p>
          </div>
        </div>
        <button
          onClick={() => logout()}
          className="mt-2 flex w-full items-center gap-2 rounded-lg px-3 py-2 text-sm text-m
        >
          <LogOut className="h-4 w-4" />
          Sign Out
        </button>
      </div>
    </aside>
  );
}

function NavItemComponent({
  item,
  pathname,
  depth = 0,
}: {
  item: NavItem;
  pathname: string;
  depth?: number;
}) {
  const isActive = pathname === item.href || pathname.startsWith(`${item.href}/`);
  const hasChildren = item.children && item.children.length > 0;
  const Icon = item.icon;

  if (hasChildren) {
    return (
```

```
    <li>
      <div
        className={cn(
          'flex cursor-pointer items-center gap-3 rounded-lg px-3 py-2 text-sm transition-
          isActive
            ? 'bg-sidebar-accent text-sidebar-accent-foreground'
            : 'text-sidebar-foreground hover:bg-sidebar-accent hover:text-sidebar-accent-f
        )}
      >
        <Icon className="h-4 w-4" />
        <span className="flex-1">{item.title}</span>
        <ChevronRight
          className={cn(
            'h-4 w-4 transition-transform',
            isActive && 'rotate-90'
          )}
        />
      </div>
      {isActive && (
        <ul className="ml-4 mt-1 space-y-1 border-l border-sidebar-border pl-2">
          {item.children.map((child) => (
            <li key={child.href}>
              <Link
                href={child.href}
                className={cn(
                  'flex items-center gap-3 rounded-lg px-3 py-2 text-sm transition-colors'
                  pathname === child.href
                    ? 'bg-sidebar-accent text-sidebar-accent-foreground'
                    : 'text-sidebar-foreground hover:bg-sidebar-accent hover:text-sidebar-
                )}
              >
                <child.icon className="h-4 w-4" />
                {child.title}
              </Link>
            </li>
          ))}
        </ul>
      )}
    </li>
  );
}

return (
  <li>
    <Link
      href={item.href}
```

```
        className={cn(
          'flex items-center gap-3 rounded-lg px-3 py-2 text-sm transition-colors',
          isActive
            ? 'bg-sidebar-accent text-sidebar-accent-foreground'
            : 'text-sidebar-foreground hover:bg-sidebar-accent hover:text-sidebar-accent-for
        )}
      >
        <Icon className="h-4 w-4" />
        <span className="flex-1">{item.title}</span>
        {item.badge !== undefined && (
          <span className="flex h-5 min-w-5 items-center justify-center rounded-full bg-prim
            {item.badge}
          </span>
        )}
      </Link>
    </li>
  );
}
```

components/layout/header.tsx

```
'use client';

import { useState } from 'react';
import { useTheme } from 'next-themes';
import { Bell, Moon, Sun, Search, Command } from 'lucide-react';
import { Button } from '@/components/ui/button';
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from '@/components/ui/dropdown-menu';
import { NotificationBell } from '@/components/notifications/notification-bell';

export function Header() {
  const { theme, setTheme } = useTheme();
  const [searchOpen, setSearchOpen] = useState(false);

  return (
    <header className="sticky top-0 z-50 flex h-16 items-center justify-between border-b bg-
      {/* Search */}
      <div className="flex items-center gap-4">
        <Button
          variant="outline"
          className="w-64 justify-start text-muted-foreground"
```

```
            onClick={() => setSearchOpen(true)}
          >
            <Search className="mr-2 h-4 w-4" />
            <span>Search...</span>
            <kbd className="pointer-events-none ml-auto inline-flex h-5 select-none items-cent
              <Command className="h-3 w-3" />K
            </kbd>
          </Button>
        </div>

        {/* Actions */}
        <div className="flex items-center gap-2">
          {/* Theme Toggle */}
          <DropdownMenu>
            <DropdownMenuTrigger asChild>
              <Button variant="ghost" size="icon">
                <Sun className="h-4 w-4 rotate-0 scale-100 transition-all dark:-rotate-90 dark
                <Moon className="absolute h-4 w-4 rotate-90 scale-0 transition-all dark:rotate
                <span className="sr-only">Toggle theme</span>
              </Button>
            </DropdownMenuTrigger>
            <DropdownMenuContent align="end">
              <DropdownMenuItem onClick={() => setTheme('light')}>
                Light
              </DropdownMenuItem>
              <DropdownMenuItem onClick={() => setTheme('dark')}>
                Dark
              </DropdownMenuItem>
              <DropdownMenuItem onClick={() => setTheme('system')}>
                System
              </DropdownMenuItem>
            </DropdownMenuContent>
          </DropdownMenu>

          {/* Notifications */}
          <NotificationBell />
        </div>
      </header>
    );
}
```

components/layout/page-header.tsx

```
import { ChevronRight, LucideIcon } from 'lucide-react';
import Link from 'next/link';
```

```
interface Breadcrumb {
  label: string;
  href?: string;
}

interface PageHeaderProps {
  title: string;
  description?: string;
  breadcrumbs?: Breadcrumb[];
  icon?: LucideIcon;
  actions?: React.ReactNode;
}

export function PageHeader({
  title,
  description,
  breadcrumbs,
  icon: Icon,
  actions,
}: PageHeaderProps) {
  return (
    <div className="mb-8">
      {/* Breadcrumbs */}
      {breadcrumbs && breadcrumbs.length > 0 && (
        <nav className="mb-4 flex items-center gap-1 text-sm text-muted-foreground">
          {breadcrumbs.map((crumb, index) => (
            <div key={crumb.label} className="flex items-center gap-1">
              {index > 0 && <ChevronRight className="h-4 w-4" />}
              {crumb.href ? (
                <Link
                  href={crumb.href}
                  className="hover:text-foreground transition-colors"
                >
                  {crumb.label}
                </Link>
              ) : (
                <span className="text-foreground">{crumb.label}</span>
              )}
            </div>
          ))}
        </nav>
      )}

      {/* Title & Actions */}
      <div className="flex items-start justify-between gap-4">
        <div className="flex items-center gap-3">
```

```
          {Icon && (
            <div className="flex h-10 w-10 items-center justify-center rounded-lg bg-primary
              <Icon className="h-5 w-5 text-primary" />
            </div>
          )}
          <div>
            <h1 className="text-2xl font-semibold tracking-tight">{title}</h1>
            {description && (
              <p className="mt-1 text-muted-foreground">{description}</p>
            )}
          </div>
        </div>
        {actions && <div className="flex items-center gap-2">{actions}</div>}
      </div>
    </div>
  );
}
```

PART 6: ROOT LAYOUT & PROVIDERS

app/layout.tsx

```
import type { Metadata, Viewport } from 'next';
import { Inter, JetBrains_Mono } from 'next/font/google';
import { ThemeProvider } from 'next-themes';
import { Toaster } from 'sonner';
import { Providers } from './providers';
import './globals.css';

const inter = Inter({
  subsets: ['latin'],
  variable: '--font-sans',
});

const jetbrainsMono = JetBrains_Mono({
  subsets: ['latin'],
  variable: '--font-mono',
});

export const metadata: Metadata = {
  title: {
    default: 'RADIANT Admin',
    template: '%s | RADIANT Admin',
  },
  description: 'Administration dashboard for RADIANT AI platform',
```

```
  icons: {
    icon: '/favicon.ico',
  },
};

export const viewport: Viewport = {
  themeColor: [
    { media: '(prefers-color-scheme: light)', color: 'white' },
    { media: '(prefers-color-scheme: dark)', color: '#09090b' },
  ],
};

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en" suppressHydrationWarning>
      <body
        className={`${inter.variable} ${jetbrainsMono.variable} font-sans antialiased`}
      >
        <ThemeProvider
          attribute="class"
          defaultTheme="system"
          enableSystem
          disableTransitionOnChange
        >
          <Providers>
            {children}
            <Toaster position="bottom-right" richColors />
          </Providers>
        </ThemeProvider>
      </body>
    </html>
  );
}
```

app/providers.tsx

```
'use client';

import { QueryClient, QueryClientProvider } from '@tanstack/react-query';
import { ReactQueryDevtools } from '@tanstack/react-query-devtools';
import { useState } from 'react';
import { AuthProvider } from '@/lib/auth/context';
```

```
import { configureAmplify } from '@/lib/auth/amplify';

// Configure Amplify on client side
if (typeof window !== 'undefined') {
  configureAmplify();
}

export function Providers({ children }: { children: React.ReactNode }) {
  const [queryClient] = useState(
    () =>
      new QueryClient({
        defaultOptions: {
          queries: {
            staleTime: 60 * 1000, // 1 minute
            gcTime: 5 * 60 * 1000, // 5 minutes
            retry: 1,
            refetchOnWindowFocus: false,
          },
        },
      })
  );

  return (
    <QueryClientProvider client={queryClient}>
      <AuthProvider>{children}</AuthProvider>
      <ReactQueryDevtools initialIsOpen={false} />
    </QueryClientProvider>
  );
}
```

app/(dashboard)/layout.tsx

```
'use client';

import { useRequireAuth } from '@/lib/auth/hooks';
import { Sidebar } from '@/components/layout/sidebar';
import { Header } from '@/components/layout/header';
import { Loader2 } from 'lucide-react';

export default function DashboardLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  const { isLoading, isAuthenticated } = useRequireAuth();
```

```
  if (isLoading) {
    return (
      <div className="flex h-screen items-center justify-center">
        <Loader2 className="h-8 w-8 animate-spin text-primary" />
      </div>
    );
  }

  if (!isAuthenticated) {
    return null; // Will redirect in useRequireAuth
  }

  return (
    <div className="flex h-screen overflow-hidden">
      <Sidebar />
      <div className="flex flex-1 flex-col overflow-hidden">
        <Header />
        <main className="flex-1 overflow-y-auto bg-muted/30 p-6">
          {children}
        </main>
      </div>
    </div>
  );
}
```

PART 7: DASHBOARD PAGE

app/(dashboard)/page.tsx

```
import { Suspense } from 'react';
import { PageHeader } from '@/components/layout/page-header';
import { LayoutDashboard } from 'lucide-react';
import { MetricsCards } from './components/metrics-cards';
import { SystemHealthCard } from './components/system-health-card';
import { ActivityFeed } from './components/activity-feed';
import { QuickActions } from './components/quick-actions';
import { UsageChart } from './components/usage-chart';
import { Skeleton } from '@/components/ui/skeleton';

export const metadata = {
  title: 'Dashboard',
};

export default function DashboardPage() {
  return (
```

```
    <div>
      <PageHeader
        title="Dashboard"
        description="Overview of your RADIANT platform"
        icon={LayoutDashboard}
      />

      {/* Metrics Grid */}
      <Suspense fallback={<MetricsSkeleton />}>
        <MetricsCards />
      </Suspense>

      {/* Main Content Grid */}
      <div className="mt-8 grid gap-6 lg:grid-cols-3">
        {/* Left Column - Charts */}
        <div className="space-y-6 lg:col-span-2">
          <Suspense fallback={<ChartSkeleton />}>
            <UsageChart />
          </Suspense>
        </div>

        {/* Right Column - Activity & Actions */}
        <div className="space-y-6">
          <Suspense fallback={<CardSkeleton />}>
            <SystemHealthCard />
          </Suspense>

          <QuickActions />

          <Suspense fallback={<CardSkeleton />}>
            <ActivityFeed />
          </Suspense>
        </div>
      </div>
    </div>
  );
}

function MetricsSkeleton() {
  return (
    <div className="grid gap-4 sm:grid-cols-2 lg:grid-cols-4">
      {[...Array(4)].map((_, i) => (
        <Skeleton key={i} className="h-32 rounded-lg" />
      ))}
    </div>
  );
```

```
}

function ChartSkeleton() {
  return <Skeleton className="h-80 rounded-lg" />;
}

function CardSkeleton() {
  return <Skeleton className="h-48 rounded-lg" />;
}
```

app/(dashboard)/components/metrics-cards.tsx

```
'use client';

import {
  ArrowUp,
  ArrowDown,
  Activity,
  Cpu,
  DollarSign,
  AlertTriangle,
} from 'lucide-react';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { cn } from '@/lib/utils/cn';
import { useQuery } from '@tanstack/react-query';
import { dashboardApi } from '@/lib/api/endpoints';

export function MetricsCards() {
  const { data: metrics, isLoading } = useQuery({
    queryKey: ['dashboard', 'metrics'],
    queryFn: () => dashboardApi.getMetrics('24h'),
  });

  if (isLoading || !metrics) {
    return null;
  }

  const cards = [
    {
      title: 'Total Requests',
      value: formatNumber(metrics.totalRequests.value),
      change: metrics.totalRequests.change,
      period: metrics.totalRequests.period,
      icon: Activity,
      color: 'text-blue-500',
      bgColor: 'bg-blue-500/10',
```

```
  },
  {
    title: 'Active Models',
    value: metrics.activeModels.value.toString(),
    subtitle: `${metrics.activeModels.external} external, ${metrics.activeModels.selfHoste
    icon: Cpu,
    color: 'text-purple-500',
    bgColor: 'bg-purple-500/10',
  },
  {
    title: 'Revenue',
    value: formatCurrency(metrics.revenue.value),
    change: metrics.revenue.change,
    period: metrics.revenue.period,
    icon: DollarSign,
    color: 'text-green-500',
    bgColor: 'bg-green-500/10',
  },
  {
    title: 'Error Rate',
    value: `${metrics.errorRate.value.toFixed(2)}%`,
    change: -metrics.errorRate.change, // Negative is good for errors
    period: metrics.errorRate.period,
    icon: AlertTriangle,
    color: metrics.errorRate.value > 1 ? 'text-red-500' : 'text-amber-500',
    bgColor: metrics.errorRate.value > 1 ? 'bg-red-500/10' : 'bg-amber-500/10',
  },
];

return (
  <div className="grid gap-4 sm:grid-cols-2 lg:grid-cols-4">
    {cards.map((card) => (
      <Card key={card.title}>
        <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
          <CardTitle className="text-sm font-medium text-muted-foreground">
            {card.title}
          </CardTitle>
          <div className={cn('rounded-md p-2', card.bgColor)}>
            <card.icon className={cn('h-4 w-4', card.color)} />
          </div>
        </CardHeader>
        <CardContent>
          <div className="text-2xl font-bold">{card.value}</div>
          {card.change !== undefined && (
            <div className="flex items-center gap-1 text-xs">
              {card.change > 0 ? (
```

```
                  <ArrowUp className="h-3 w-3 text-green-500" />
                ) : (
                  <ArrowDown className="h-3 w-3 text-red-500" />
                )}
                <span
                  className={cn(
                    card.change > 0 ? 'text-green-500' : 'text-red-500'
                  )}
                >
                  {Math.abs(card.change).toFixed(1)}%
                </span>
                <span className="text-muted-foreground">
                  vs last {card.period}
                </span>
              </div>
            )}
            {card.subtitle && (
              <p className="mt-1 text-xs text-muted-foreground">
                {card.subtitle}
              </p>
            )}
          </CardContent>
        </Card>
      ))}
    </div>
  );
}

function formatNumber(num: number): string {
  if (num >= 1_000_000) {
    return `${(num / 1_000_000).toFixed(1)}M`;
  }
  if (num >= 1_000) {
    return `${(num / 1_000).toFixed(1)}K`;
  }
  return num.toString();
}

function formatCurrency(amount: number): string {
  return new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency: 'USD',
    minimumFractionDigits: 0,
    maximumFractionDigits: 0,
  }).format(amount);
}
```

app/(dashboard)/components/system-health-card.tsx

```
'use client';

import { useQuery } from '@tanstack/react-query';
import { CheckCircle2, AlertCircle, XCircle, RefreshCw } from 'lucide-react';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Button } from '@/components/ui/button';
import { cn } from '@/lib/utils/cn';
import { dashboardApi } from '@/lib/api/endpoints';

export function SystemHealthCard() {
  const { data: health, isLoading, refetch, isFetching } = useQuery({
    queryKey: ['dashboard', 'health'],
    queryFn: () => dashboardApi.getHealth(),
    refetchInterval: 30000, // Refresh every 30 seconds
  });

  const statusConfig = {
    healthy: {
      icon: CheckCircle2,
      color: 'text-green-500',
      bgColor: 'bg-green-500/10',
      label: 'All Systems Operational',
    },
    degraded: {
      icon: AlertCircle,
      color: 'text-amber-500',
      bgColor: 'bg-amber-500/10',
      label: 'Some Systems Degraded',
    },
    critical: {
      icon: XCircle,
      color: 'text-red-500',
      bgColor: 'bg-red-500/10',
      label: 'System Issues Detected',
    },
  };

  const status = health?.overall || 'healthy';
  const config = statusConfig[status];
  const StatusIcon = config.icon;

  return (
    <Card>
      <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
```

```
  <CardTitle className="text-sm font-medium">System Health</CardTitle>
  <Button
    variant="ghost"
    size="icon"
    onClick={() => refetch()}
    disabled={isFetching}
  >
    <RefreshCw
      className={cn('h-4 w-4', isFetching && 'animate-spin')}
    />
  </Button>
</CardHeader>
<CardContent>
  {isLoading ? (
    <div className="space-y-2">
      <div className="h-10 animate-pulse rounded bg-muted" />
      <div className="space-y-1">
        {[...Array(4)].map((_, i) => (
          <div key={i} className="h-6 animate-pulse rounded bg-muted" />
        ))}
      </div>
    </div>
  ) : (
    <>
      <div
        className={cn(
          'mb-4 flex items-center gap-2 rounded-lg p-3',
          config.bgColor
        )}
      >
        <StatusIcon className={cn('h-5 w-5', config.color)} />
        <span className={cn('font-medium', config.color)}>
          {config.label}
        </span>
      </div>

      <div className="space-y-2">
        {health?.components.map((component) => (
          <div
            key={component.name}
            className="flex items-center justify-between text-sm"
          >
            <span className="text-muted-foreground">
              {component.name}
            </span>
            <ComponentStatus status={component.status} />
```

```
              </div>
            ))}
          </div>
        </>
      )}
    </CardContent>
  </Card>
);
}

function ComponentStatus({ status }: { status: 'healthy' | 'degraded' | 'down' }) {
  const configs = {
    healthy: { color: 'text-green-500', bg: 'bg-green-500', label: 'Healthy' },
    degraded: { color: 'text-amber-500', bg: 'bg-amber-500', label: 'Degraded' },
    down: { color: 'text-red-500', bg: 'bg-red-500', label: 'Down' },
  };

  const config = configs[status];

  return (
    <div className="flex items-center gap-2">
      <div className={cn('h-2 w-2 rounded-full', config.bg)} />
      <span className={cn('text-xs font-medium', config.color)}>
        {config.label}
      </span>
    </div>
  );
}
```

app/(dashboard)/components/quick-actions.tsx

```
'use client';

import Link from 'next/link';
import {
  Plus,
  Settings,
  Users,
  Shield,
  BarChart3,
  ArrowRight,
} from 'lucide-react';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Button } from '@/components/ui/button';
import { useAuth } from '@/lib/auth/context';
```

```
const actions = [
  {
    label: 'Invite Admin',
    href: '/administrators/invitations',
    icon: Users,
    permission: 'admin:write',
  },
  {
    label: 'Manage Models',
    href: '/models',
    icon: Settings,
  },
  {
    label: 'View Approvals',
    href: '/administrators/approvals',
    icon: Shield,
    permission: 'approvals:read',
  },
  {
    label: 'View Usage',
    href: '/usage',
    icon: BarChart3,
  },
];

export function QuickActions() {
  const { hasPermission } = useAuth();

  const filteredActions = actions.filter(
    (action) => !action.permission || hasPermission(action.permission)
  );

  return (
    <Card>
      <CardHeader>
        <CardTitle className="text-sm font-medium">Quick Actions</CardTitle>
      </CardHeader>
      <CardContent className="grid gap-2">
        {filteredActions.map((action) => (
          <Button
            key={action.label}
            variant="ghost"
            className="justify-between"
            asChild
          >
            <Link href={action.href}>
```

```
              <span className="flex items-center gap-2">
                <action.icon className="h-4 w-4" />
                {action.label}
              </span>
              <ArrowRight className="h-4 w-4 text-muted-foreground" />
            </Link>
          </Button>
        ))}
      </CardContent>
    </Card>
  );
}
```

PART 8: MODELS PAGE

app/(dashboard)/models/page.tsx

```
import { Suspense } from 'react';
import { PageHeader } from '@/components/layout/page-header';
import { Cpu, Plus } from 'lucide-react';
import { Button } from '@/components/ui/button';
import { ModelsContent } from './components/models-content';
import { Skeleton } from '@/components/ui/skeleton';

export const metadata = {
  title: 'AI Models',
};

export default function ModelsPage() {
  return (
    <div>
      <PageHeader
        title="AI Models"
        description="Manage external and self-hosted AI models"
        icon={Cpu}
        breadcrumbs={[
          { label: 'Dashboard', href: '/' },
          { label: 'AI Models' },
        ]}
      />

      <Suspense fallback={<ModelsPageSkeleton />}>
        <ModelsContent />
      </Suspense>
    </div>
```

```
  );
}

function ModelsPageSkeleton() {
  return (
    <div className="space-y-4">
      <div className="flex gap-4">
        <Skeleton className="h-10 w-64" />
        <Skeleton className="h-10 w-32" />
        <Skeleton className="h-10 w-32" />
      </div>
      <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">
        {[...Array(6)].map((_, i) => (
          <Skeleton key={i} className="h-48 rounded-lg" />
        ))}
      </div>
    </div>
  );
}
```

app/(dashboard)/models/components/models-content.tsx

```
'use client';

import { useState } from 'react';
import { useModels } from '@/lib/hooks/use-models';
import { ModelCard } from '@/components/models/model-card';
import { ModelFilters } from '@/components/models/model-filters';
import { Input } from '@/components/ui/input';
import { Search, Grid3X3, List } from 'lucide-react';
import { Button } from '@/components/ui/button';
import { Tabs, TabsList, TabsTrigger } from '@/components/ui/tabs';
import type { ModelFilters as ModelFiltersType } from '@/lib/api/types';
import { cn } from '@/lib/utils/cn';

export function ModelsContent() {
  const [filters, setFilters] = useState<ModelFiltersType>({});
  const [searchQuery, setSearchQuery] = useState('');
  const [viewMode, setViewMode] = useState<'grid' | 'list'>('grid');

  const { data, isLoading } = useModels({
    ...filters,
    search: searchQuery || undefined,
  });

  const models = data?.data || [];
```

```
return (
  <div className="space-y-6">
    {/* Filters Bar */}
    <div className="flex flex-wrap items-center gap-4">
      <div className="relative flex-1 min-w-[200px] max-w-md">
        <Search className="absolute left-3 top-1/2 h-4 w-4 -translate-y-1/2 text-muted-for
        <Input
          placeholder="Search models..."
          value={searchQuery}
          onChange={(e) => setSearchQuery(e.target.value)}
          className="pl-9"
        />
      </div>

      <ModelFilters filters={filters} onFiltersChange={setFilters} />

      <div className="ml-auto flex items-center gap-2">
        <Button
          variant={viewMode === 'grid' ? 'secondary' : 'ghost'}
          size="icon"
          onClick={() => setViewMode('grid')}
        >
          <Grid3X3 className="h-4 w-4" />
        </Button>
        <Button
          variant={viewMode === 'list' ? 'secondary' : 'ghost'}
          size="icon"
          onClick={() => setViewMode('list')}
        >
          <List className="h-4 w-4" />
        </Button>
      </div>
    </div>

    {/* Tabs for model types */}
    <Tabs defaultValue="all" className="space-y-4">
      <TabsList>
        <TabsTrigger value="all">All Models</TabsTrigger>
        <TabsTrigger value="external">External</TabsTrigger>
        <TabsTrigger value="self-hosted">Self-Hosted</TabsTrigger>
      </TabsList>
    </Tabs>

    {/* Models Grid/List */}
    {isLoading ? (
```

```
        <div className="text-center py-8 text-muted-foreground">
          Loading models...
        </div>
      ) : models.length === 0 ? (
        <div className="text-center py-8 text-muted-foreground">
          No models found matching your criteria
        </div>
      ) : (
        <div
          className={cn(
            viewMode === 'grid'
              ? 'grid gap-4 md:grid-cols-2 lg:grid-cols-3'
              : 'space-y-4'
          )}
        >
          {models.map((model) => (
            <ModelCard key={model.id} model={model} compact={viewMode === 'list'} />
          ))}
        </div>
      )}
    </div>
  );
}
```

components/models/model-card.tsx

```
'use client';

import Link from 'next/link';
import {
  MoreVertical,
  ExternalLink,
  Settings,
  Power,
  Flame,
  Snowflake,
  Thermometer,
  Zap,
  Bot,
} from 'lucide-react';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Badge } from '@/components/ui/badge';
import { Button } from '@/components/ui/button';
import {
  DropdownMenu,
  DropdownMenuContent,
```

```
  DropdownMenuItem,
  DropdownMenuSeparator,
  DropdownMenuTrigger,
} from '@/components/ui/dropdown-menu';
import { ThermalBadge } from './thermal-badge';
import type { Model } from '@/lib/api/types';
import { cn } from '@/lib/utils/cn';
import { useSetModelEnabled, useWarmUpModel } from '@/lib/hooks/use-models';
import { toast } from 'sonner';

interface ModelCardProps {
  model: Model;
  compact?: boolean;
}

export function ModelCard({ model, compact = false }: ModelCardProps) {
  const { mutate: setEnabled, isPending: isTogglingEnabled } = useSetModelEnabled();
  const { mutate: warmUp, isPending: isWarmingUp } = useWarmUpModel();

  const handleToggleEnabled = () => {
    setEnabled(
      { id: model.id, enabled: !model.isEnabled },
      {
        onSuccess: () => {
          toast.success(
            model.isEnabled
              ? `${model.displayName} disabled`
              : `${model.displayName} enabled`
          );
        },
        onError: () => {
          toast.error('Failed to update model');
        },
      }
    );
  };

  const handleWarmUp = () => {
    warmUp(model.id, {
      onSuccess: (data) => {
        toast.success(
          `Warming up ${model.displayName}. Estimated time: ${data.estimatedWaitSeconds}s`
        );
      },
      onError: () => {
        toast.error('Failed to warm up model');
```

```
    },
  });
};

if (compact) {
  return (
    <Card className="flex items-center gap-4 p-4">
      <div className="flex h-10 w-10 items-center justify-center rounded-lg bg-primary/10'
        <Bot className="h-5 w-5 text-primary" />
      </div>
      <div className="flex-1 min-w-0">
        <div className="flex items-center gap-2">
          <h3 className="font-medium truncate">{model.displayName}</h3>
          {model.isExternal ? (
            <Badge variant="outline" className="text-xs">External</Badge>
          ) : (
            <Badge variant="secondary" className="text-xs">Self-Hosted</Badge>
          )}
        </div>
        <p className="text-sm text-muted-foreground truncate">
          {model.provider} Ã¸Â· {model.category.replace('_', ' ')}
        </p>
      </div>
      <ThermalBadge state={model.thermalState} />
      <Badge variant={model.isEnabled ? 'default' : 'secondary'}>
        {model.isEnabled ? 'Enabled' : 'Disabled'}
      </Badge>
      <Button variant="ghost" size="sm" asChild>
        <Link href={`/models/${model.id}`}>
          <Settings className="h-4 w-4" />
        </Link>
      </Button>
    </Card>
  );
}

return (
  <Card>
    <CardHeader className="flex flex-row items-start justify-between space-y-0">
      <div className="flex items-center gap-3">
        <div className="flex h-10 w-10 items-center justify-center rounded-lg bg-primary/1
          <Bot className="h-5 w-5 text-primary" />
        </div>
        <div>
          <CardTitle className="text-base">{model.displayName}</CardTitle>
          <p className="text-xs text-muted-foreground">{model.provider}</p>
```

```
    </div>
  </div>
  <DropdownMenu>
    <DropdownMenuTrigger asChild>
      <Button variant="ghost" size="icon">
        <MoreVertical className="h-4 w-4" />
      </Button>
    </DropdownMenuTrigger>
    <DropdownMenuContent align="end">
      <DropdownMenuItem asChild>
        <Link href={`/models/${model.id}`}>
          <Settings className="mr-2 h-4 w-4" />
          Configure
        </Link>
      </DropdownMenuItem>
      {!model.isExternal && model.thermalState === 'COLD' && (
        <DropdownMenuItem onClick={handleWarmUp} disabled={isWarmingUp}>
          <Flame className="mr-2 h-4 w-4" />
          Warm Up
        </DropdownMenuItem>
      )}
      <DropdownMenuSeparator />
      <DropdownMenuItem
        onClick={handleToggleEnabled}
        disabled={isTogglingEnabled}
      >
        <Power className="mr-2 h-4 w-4" />
        {model.isEnabled ? 'Disable' : 'Enable'}
      </DropdownMenuItem>
    </DropdownMenuContent>
  </DropdownMenu>
</CardHeader>
<CardContent className="space-y-4">
  <p className="text-sm text-muted-foreground line-clamp-2">
    {model.description}
  </p>

  <div className="flex flex-wrap gap-2">
    {model.isExternal ? (
      <Badge variant="outline">External</Badge>
    ) : (
      <Badge variant="secondary">Self-Hosted</Badge>
    )}
    <Badge variant="outline" className="capitalize">
      {model.category.replace('_', ' ')}
    </Badge>
```

```
          <ThermalBadge state={model.thermalState} />
        </div>

        <div className="grid grid-cols-2 gap-4 text-sm">
          <div>
            <p className="text-muted-foreground">Status</p>
            <p className={cn(
              'font-medium',
              model.isEnabled ? 'text-green-500' : 'text-muted-foreground'
            )}>
              {model.isEnabled ? 'Enabled' : 'Disabled'}
            </p>
          </div>
          <div>
            <p className="text-muted-foreground">Requests (24h)</p>
            <p className="font-medium">{model.usage.last24h.toLocaleString()}</p>
          </div>
        </div>
      </CardContent>
    </Card>
  );
}
```

components/models/thermal-badge.tsx

```
import { Flame, Snowflake, Thermometer, Zap, Settings } from 'lucide-react';
import { Badge } from '@/components/ui/badge';
import type { ThermalState } from '@/lib/api/types';
import { cn } from '@/lib/utils/cn';

interface ThermalBadgeProps {
  state: ThermalState;
  showLabel?: boolean;
}

const thermalConfig: Record<ThermalState, {
  icon: React.ComponentType<{ className?: string }>;
  label: string;
  className: string;
}> = {
  OFF: {
    icon: Settings,
    label: 'Off',
    className: 'bg-gray-500/10 text-gray-500 border-gray-500/20',
  },
  COLD: {
```

```
    icon: Snowflake,
    label: 'Cold',
    className: 'bg-blue-500/10 text-blue-500 border-blue-500/20',
  },
  WARM: {
    icon: Thermometer,
    label: 'Warm',
    className: 'bg-amber-500/10 text-amber-500 border-amber-500/20',
  },
  HOT: {
    icon: Flame,
    label: 'Hot',
    className: 'bg-red-500/10 text-red-500 border-red-500/20',
  },
  AUTOMATIC: {
    icon: Zap,
    label: 'Auto',
    className: 'bg-violet-500/10 text-violet-500 border-violet-500/20',
  },
};

export function ThermalBadge({ state, showLabel = true }: ThermalBadgeProps) {
  const config = thermalConfig[state];
  const Icon = config.icon;

  return (
    <Badge variant="outline" className={cn('gap-1', config.className)}>
      <Icon className="h-3 w-3" />
      {showLabel && config.label}
    </Badge>
  );
}
```

PART 9: ADMINISTRATORS & APPROVALS PAGES

app/(dashboard)/administrators/page.tsx

```
import { Suspense } from 'react';
import { PageHeader } from '@/components/layout/page-header';
import { Users, UserPlus } from 'lucide-react';
import { Button } from '@/components/ui/button';
import Link from 'next/link';
import { AdminTable } from '@/components/administrators/admin-table';
import { Skeleton } from '@/components/ui/skeleton';
```

```
export const metadata = {
  title: 'Administrators',
};

export default function AdministratorsPage() {
  return (
    <div>
      <PageHeader
        title="Administrators"
        description="Manage platform administrators and their roles"
        icon={Users}
        breadcrumbs={[
          { label: 'Dashboard', href: '/' },
          { label: 'Administrators' },
        ]}
        actions={
          <Button asChild>
            <Link href="/administrators/invitations">
              <UserPlus className="mr-2 h-4 w-4" />
              Invite Admin
            </Link>
          </Button>
        }
      />

      <Suspense fallback={<Skeleton className="h-96" />}>
        <AdminTable />
      </Suspense>
    </div>
  );
}
```

app/(dashboard)/administrators/approvals/page.tsx

```
import { Suspense } from 'react';
import { PageHeader } from '@/components/layout/page-header';
import { ClipboardCheck } from 'lucide-react';
import { ApprovalsContent } from './components/approvals-content';
import { Skeleton } from '@/components/ui/skeleton';

export const metadata = {
  title: 'Approval Queue',
};

export default function ApprovalsPage() {
  return (
```

```
    <div>
      <PageHeader
        title="Approval Queue"
        description="Review and approve production deployment requests"
        icon={ClipboardCheck}
        breadcrumbs={[
          { label: 'Dashboard', href: '/' },
          { label: 'Administrators', href: '/administrators' },
          { label: 'Approvals' },
        ]}
      />

      <Suspense fallback={<Skeleton className="h-96" />}>
        <ApprovalsContent />
      </Suspense>
    </div>
  );
}
```

app/(dashboard)/administrators/approvals/components/approvals-content.tsx

```
'use client';

import { useState } from 'react';
import { useApprovals, useApproveRequest, useRejectRequest } from '@/lib/hooks/use-administr
import { ApprovalCard } from '@/components/administrators/approval-card';
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';
import { toast } from 'sonner';
import type { ApprovalRequest } from '@/lib/api/types';

export function ApprovalsContent() {
  const [status, setStatus] = useState<'pending' | 'approved' | 'rejected'>('pending');

  const { data: approvals, isLoading } = useApprovals(status);
  const { mutate: approve, isPending: isApproving } = useApproveRequest();
  const { mutate: reject, isPending: isRejecting } = useRejectRequest();

  const handleApprove = (id: string, notes?: string) => {
    approve(
      { id, notes },
      {
        onSuccess: () => {
          toast.success('Request approved');
        },
        onError: () => {
```

```
        toast.error('Failed to approve request');
      },
    }
  );
};

const handleReject = (id: string, reason: string) => {
  reject(
    { id, reason },
    {
      onSuccess: () => {
        toast.success('Request rejected');
      },
      onError: () => {
        toast.error('Failed to reject request');
      },
    }
  );
};

return (
  <Tabs value={status} onValueChange={(v) => setStatus(v as typeof status)}>
    <TabsList>
      <TabsTrigger value="pending">Pending</TabsTrigger>
      <TabsTrigger value="approved">Approved</TabsTrigger>
      <TabsTrigger value="rejected">Rejected</TabsTrigger>
    </TabsList>

    <TabsContent value={status} className="mt-6">
      {isLoading ? (
        <div className="text-center py-8 text-muted-foreground">
          Loading approvals...
        </div>
      ) : !approvals || approvals.length === 0 ? (
        <div className="text-center py-8 text-muted-foreground">
          No {status} approval requests
        </div>
      ) : (
        <div className="space-y-4">
          {approvals.map((approval) => (
            <ApprovalCard
              key={approval.id}
              approval={approval}
              onApprove={handleApprove}
              onReject={handleReject}
              isApproving={isApproving}
```

```
                isRejecting={isRejecting}
                showActions={status === 'pending'}
              />
            ))}
          </div>
        )}
      </TabsContent>
    </Tabs>
  );
}
```

components/administrators/approval-card.tsx

```
'use client';

import { useState } from 'react';
import { format, formatDistanceToNow } from 'date-fns';
import {
  Clock,
  AlertTriangle,
  CheckCircle2,
  XCircle,
  ChevronDown,
  ChevronUp,
  Shield,
} from 'lucide-react';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Badge } from '@/components/ui/badge';
import { Button } from '@/components/ui/button';
import { Textarea } from '@/components/ui/textarea';
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
  DialogTrigger,
} from '@/components/ui/dialog';
import type { ApprovalRequest } from '@/lib/api/types';
import { cn } from '@/lib/utils/cn';

interface ApprovalCardProps {
  approval: ApprovalRequest;
  onApprove: (id: string, notes?: string) => void;
  onReject: (id: string, reason: string) => void;
```

```
  isApproving: boolean;
  isRejecting: boolean;
  showActions?: boolean;
}

const riskColors = {
  low: 'bg-green-500/10 text-green-500 border-green-500/20',
  medium: 'bg-amber-500/10 text-amber-500 border-amber-500/20',
  high: 'bg-orange-500/10 text-orange-500 border-orange-500/20',
  critical: 'bg-red-500/10 text-red-500 border-red-500/20',
};

const statusIcons = {
  pending: Clock,
  approved: CheckCircle2,
  rejected: XCircle,
  expired: AlertTriangle,
  executed: CheckCircle2,
};

export function ApprovalCard({
  approval,
  onApprove,
  onReject,
  isApproving,
  isRejecting,
  showActions = true,
}: ApprovalCardProps) {
  const [expanded, setExpanded] = useState(false);
  const [rejectReason, setRejectReason] = useState('');
  const [approvalNotes, setApprovalNotes] = useState('');
  const [showRejectDialog, setShowRejectDialog] = useState(false);

  const StatusIcon = statusIcons[approval.status];

  return (
    <Card>
      <CardHeader className="flex flex-row items-start justify-between space-y-0">
        <div className="flex-1">
          <div className="flex items-center gap-2">
            <Shield className="h-5 w-5 text-muted-foreground" />
            <CardTitle className="text-base">{approval.title}</CardTitle>
          </div>
          <p className="mt-1 text-sm text-muted-foreground">
            {approval.description}
          </p>
```

```
  </div>
  <div className="flex items-center gap-2">
    <Badge variant="outline" className={cn(riskColors[approval.riskLevel])}>
      {approval.riskLevel.toUpperCase()} RISK
    </Badge>
    <Badge variant="outline" className="capitalize">
      {approval.environment}
    </Badge>
  </div>
</CardHeader>

<CardContent className="space-y-4">
  {/* Meta Info */}
  <div className="grid grid-cols-2 gap-4 text-sm md:grid-cols-4">
    <div>
      <p className="text-muted-foreground">Type</p>
      <p className="font-medium capitalize">
        {approval.type.replace(/_/g, ' ')}
      </p>
    </div>
    <div>
      <p className="text-muted-foreground">Requested By</p>
      <p className="font-medium">{approval.initiatedBy.name}</p>
    </div>
    <div>
      <p className="text-muted-foreground">Requested</p>
      <p className="font-medium">
        {formatDistanceToNow(new Date(approval.initiatedAt), {
          addSuffix: true,
        })}
      </p>
    </div>
    <div>
      <p className="text-muted-foreground">Expires</p>
      <p className="font-medium">
        {format(new Date(approval.expiresAt), 'MMM d, h:mm a')}
      </p>
    </div>
  </div>

  {/* Expandable Details */}
  <Button
    variant="ghost"
    className="w-full justify-between"
    onClick={() => setExpanded(!expanded)}
  >
```

```
  <span>View Details</span>
  {expanded ? (
    <ChevronUp className="h-4 w-4" />
  ) : (
    <ChevronDown className="h-4 w-4" />
  )}
</Button>

{expanded && (
  <div className="rounded-lg bg-muted p-4 text-sm">
    <pre className="whitespace-pre-wrap">
      {JSON.stringify(approval.payload, null, 2)}
    </pre>
  </div>
)}

{/* Actions */}
{showActions && approval.status === 'pending' && (
  <div className="flex gap-2 pt-4 border-t">
    <Dialog>
      <DialogTrigger asChild>
        <Button className="flex-1" disabled={isApproving}>
          <CheckCircle2 className="mr-2 h-4 w-4" />
          Approve
        </Button>
      </DialogTrigger>
      <DialogContent>
        <DialogHeader>
          <DialogTitle>Approve Request</DialogTitle>
          <DialogDescription>
            You are about to approve this {approval.type.replace(/_/g, ' ')} request
            for the {approval.environment} environment. This action cannot be undone
          </DialogDescription>
        </DialogHeader>
        <Textarea
          placeholder="Optional approval notes..."
          value={approvalNotes}
          onChange={(e) => setApprovalNotes(e.target.value)}
        />
        <DialogFooter>
          <Button
            onClick={() => onApprove(approval.id, approvalNotes)}
            disabled={isApproving}
          >
            Confirm Approval
          </Button>
```

```
                </DialogFooter>
              </DialogContent>
            </Dialog>

            <Dialog open={showRejectDialog} onOpenChange={setShowRejectDialog}>
              <DialogTrigger asChild>
                <Button
                  variant="outline"
                  className="flex-1"
                  disabled={isRejecting}
                >
                  <XCircle className="mr-2 h-4 w-4" />
                  Reject
                </Button>
              </DialogTrigger>
              <DialogContent>
                <DialogHeader>
                  <DialogTitle>Reject Request</DialogTitle>
                  <DialogDescription>
                    Please provide a reason for rejecting this request.
                  </DialogDescription>
                </DialogHeader>
                <Textarea
                  placeholder="Reason for rejection (required)..."
                  value={rejectReason}
                  onChange={(e) => setRejectReason(e.target.value)}
                />
                <DialogFooter>
                  <Button
                    variant="destructive"
                    onClick={() => {
                      onReject(approval.id, rejectReason);
                      setShowRejectDialog(false);
                    }}
                    disabled={isRejecting || !rejectReason.trim()}
                  >
                    Confirm Rejection
                  </Button>
                </DialogFooter>
              </DialogContent>
            </Dialog>
          </div>
        )}
      </CardContent>
    </Card>
  );
```

```
}
```

PART 10: UTILITY FUNCTIONS

lib/utils/cn.ts

```typescript
import { type ClassValue, clsx } from 'clsx';
import { twMerge } from 'tailwind-merge';

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs));
}
```

lib/utils/format.ts

```typescript
import { format, formatDistanceToNow } from 'date-fns';

export function formatNumber(num: number): string {
  if (num >= 1_000_000_000) {
    return `${(num / 1_000_000_000).toFixed(1)}B`;
  }
  if (num >= 1_000_000) {
    return `${(num / 1_000_000).toFixed(1)}M`;
  }
  if (num >= 1_000) {
    return `${(num / 1_000).toFixed(1)}K`;
  }
  return num.toLocaleString();
}

export function formatCurrency(amount: number, currency = 'USD'): string {
  return new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency,
    minimumFractionDigits: 0,
    maximumFractionDigits: 2,
  }).format(amount);
}

export function formatBytes(bytes: number): string {
  if (bytes === 0) return '0 B';
  const k = 1024;
  const sizes = ['B', 'KB', 'MB', 'GB', 'TB'];
  const i = Math.floor(Math.log(bytes) / Math.log(k));
  return `${parseFloat((bytes / Math.pow(k, i)).toFixed(2))} ${sizes[i]}`;
```

```
}

export function formatDate(date: string | Date): string {
  return format(new Date(date), 'MMM d, yyyy');
}

export function formatDateTime(date: string | Date): string {
  return format(new Date(date), 'MMM d, yyyy h:mm a');
}

export function formatRelativeTime(date: string | Date): string {
  return formatDistanceToNow(new Date(date), { addSuffix: true });
}

export function formatPercentage(value: number, decimals = 1): string {
  return `${value.toFixed(decimals)}%`;
}

export function formatTokens(tokens: number): string {
  if (tokens >= 1_000_000) {
    return `${(tokens / 1_000_000).toFixed(2)}M tokens`;
  }
  if (tokens >= 1_000) {
    return `${(tokens / 1_000).toFixed(1)}K tokens`;
  }
  return `${tokens} tokens`;
}
```

lib/utils/constants.ts

```
export const APP_NAME = 'RADIANT Admin';
export const APP_VERSION = '2.2.0';

export const THERMAL_STATES = ['OFF', 'COLD', 'WARM', 'HOT', 'AUTOMATIC'] as const;
export const SERVICE_STATES = ['RUNNING', 'DEGRADED', 'DISABLED', 'OFFLINE'] as const;

export const ADMIN_ROLES = {
  SUPER_ADMIN: 'super_admin',
  ADMIN: 'admin',
  OPERATOR: 'operator',
  AUDITOR: 'auditor',
} as const;

export const ROLE_LABELS: Record<string, string> = {
  super_admin: 'Super Admin',
  admin: 'Admin',
```

```
  operator: 'Operator',
  auditor: 'Auditor',
};

export const ENVIRONMENT_COLORS: Record<string, string> = {
  dev: 'bg-blue-500',
  staging: 'bg-amber-500',
  prod: 'bg-red-500',
};

export const CHART_COLORS = [
  'hsl(262, 83%, 58%)',  // Primary purple
  'hsl(173, 80%, 40%)',  // Teal
  'hsl(197, 37%, 24%)',  // Navy
  'hsl(43, 74%, 66%)',   // Gold
  'hsl(27, 87%, 67%)',   // Orange
];
```

DEPLOYMENT

Building for Production

```
cd apps/admin-dashboard

# Install dependencies
pnpm install

# Build for production
pnpm build

# The output will be in .next/standalone
```

CDK Stack Integration

The admin dashboard is deployed via the AdminStack created in Prompt 3. The stack:

1. Creates an S3 bucket for static hosting
2. Deploys a CloudFront distribution with:
   - Security headers (CSP, X-Frame-Options, etc.)
   - TLS 1.3 with custom certificate
   - Geographic restriction (optional)
3. Syncs the built Next.js output to S3

NEXT PROMPTS

Continue with: - **Prompt 9**: Assembly & Deployment Guide

End of Prompt 8: Admin Web Dashboard (Next.js) RADIANT v2.2.0 - December 2024

â• â•

END OF SECTION 8

â• â•

â• â•

SECTION-09-DEPLOYMENT-GUIDE

SECTION 9: ASSEMBLY & DEPLOYMENT GUIDE (v2.2.0)

â• â•

> **Dependencies:** ALL previous sections **Provides:** Verification steps, testing procedures, troubleshooting

---

RADIANT v2.2.0 - Prompt 9: Assembly & Deployment Guide

> **Prompt 9 of 9** | Target Size: ~15KB | Version: 3.7.0 | December 2024

OVERVIEW

This is the final prompt in the 9-part RADIANT series. This prompt provides:

1. **Project Assembly** - How to combine all components from Prompts 1-8

2. **Deployment Checklist** - Pre-deployment, deployment, and post-deployment steps
3. **Testing Procedures** - Integration tests, smoke tests, compliance verification
4. **Success Criteria** - What constitutes a successful deployment
5. **Troubleshooting Guide** - Common issues and solutions
6. **Version History** - Changelog and upgrade notes

PROMPT SERIES RECAP

Prompt	Component	Est. Size	Key Deliverables
1	Foundation & Swift App	~50KB	Monorepo structure, Swift macOS deployment app
2	CDK Infrastructure	~45KB	VPC, Aurora, DynamoDB, S3, KMS, WAF
3	CDK AI & API Stacks	~50KB	Cognito, LiteLLM, API Gateway, AppSync
4	Lambda Core	~60KB	Router, Chat, Models, Providers, PHI
5	Lambda Admin & Billing	~55KB	Invitations, Approvals, Metering, Billing
6	Self-Hosted Models	~75KB	30+ SageMaker models, Thermal states, Mid-level services
7	External Providers & DB	~85KB	21 providers, PostgreSQL schema, DynamoDB tables
8	Admin Dashboard	~85KB	Next.js 14 dashboard, all management UIs
9	**Assembly & Deployment**	**~15KB**	**This guide**

Total Implementation: ~520KB of implementation prompts

PART 1: PROJECT ASSEMBLY

1.1 Directory Structure After All Prompts

After executing Prompts 1-8 in sequence, your project should have this structure:

```
radiant/
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ README.md
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ package.json
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ pnpm-workspace.yaml
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ tsconfig.base.json
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ .gitignore
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ .nvmrc
Ã¢â€â€š
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ packages/
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ shared/                              # From Prompt 1
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ package.json
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ tsconfig.json
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ src/
Ã¢â€â€š   Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š   Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ types/
Ã¢â€â€š   Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ constants/
Ã¢â€â€š   Ã¢â€â€š       Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ utils/
Ã¢â€â€š   Ã¢â€â€š
Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ infrastructure/                   # From Prompts 2-3
Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ package.json
Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ tsconfig.json
Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ cdk.json
Ã¢â€â€š       Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ bin/
Ã¢â€â€š       Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ radiant.ts
Ã¢â€â€š       Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ lib/
Ã¢â€â€š           Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ stacks/
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ foundation-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ networking-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ security-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ data-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ storage-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ auth-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ ai-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ api-stack.ts
Ã¢â€â€š           Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ admin-stack.ts
Ã¢â€â€š           Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ constructs/
Ã¢â€â€š
Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ functions/                              # From Prompts 4-5
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ router/
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ chat/
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ models/
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ index.ts
Ã¢â€â€š   Ã¢â€Å"Ã¢â€â¬Ã¢â€â¬ providers/
Ã¢â€â€š   Ã¢â€â€š   Ã¢â€â€Ã¢â€â¬Ã¢â€â¬ index.ts
```

```
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ phi-service/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ admin-invitations/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ admin-approvals/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ billing-metering/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ index.ts
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ thermal-manager/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ index.ts
Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ model-sync/
Ã¢â€ â€š       Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ index.ts
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ config/                                  # From Prompts 6-7
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ litellm/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ config.yaml
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ self-hosted-config.yaml
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ models/
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ external-providers.json
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ self-hosted-models.json
Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ sagemaker/
Ã¢â€ â€š       Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ endpoint-configs/
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ migrations/                              # From Prompt 7
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ 001_initial_schema.sql
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ 002_rls_policies.sql
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ 003_model_registry.sql
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ 004_billing_tables.sql
Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ 005_admin_tables.sql
Ã¢â€ â€š
Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ apps/
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ swift-deployer/                  # From Prompt 1
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ RadiantAdmin.xcodeproj
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ RadiantAdmin/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ App/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ Views/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ Services/
Ã¢â€ â€š   Ã¢â€ â€š       Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ Models/
Ã¢â€ â€š   Ã¢â€ â€š
Ã¢â€ â€š   Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ admin-dashboard/                   # From Prompt 8
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ package.json
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ next.config.js
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ tailwind.config.js
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ app/
Ã¢â€ â€š       Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ components/
Ã¢â€ â€š       Ã¢â€ â€Ã¢â€ â¸¬Ã¢â€ â¸¬ lib/
```

```
Ã¢â€ â€š
Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ docs/
    Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ architecture.md
    Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ api-reference.md
    Ã¢â€ Å"Ã¢â€ â¸¬Ã¢â€ â¸¬ deployment-guide.md
    Ã¢â€ â€ Ã¢â€ â¸¬Ã¢â€ â¸¬ troubleshooting.md
```

1.2 Assembly Verification

Run these commands to verify all components are in place:

```bash
# Navigate to project root
cd radiant

# Verify structure
echo "Checking project structure..."
ls -la packages/shared/src/types/
ls -la packages/infrastructure/lib/stacks/
ls -la functions/
ls -la apps/swift-deployer/
ls -la apps/admin-dashboard/app/
ls -la migrations/

# Install dependencies
pnpm install

# Build shared package
cd packages/shared && pnpm build && cd ../..

# Verify CDK synthesizes
cd packages/infrastructure
npx cdk synth --context environment=dev --context tier=1
cd ../..

# Build admin dashboard
cd apps/admin-dashboard && pnpm build && cd ../..

echo "Assembly verification complete!"
```

PART 2: DEPLOYMENT CHECKLIST

2.1 Pre-Deployment Requirements

AWS Account Setup

Requirement	Action	Verification
AWS Account	Create or use existing	Account ID available
IAM User	Create with AdministratorAccess	Access keys generated
AWS CLI	Install and configure	`aws sts get-caller-identity`
Route 53 Domain (optional)	Register or transfer domain	Domain in hosted zone
ACM Certificate (optional)	Request in us-east-1	Certificate validated
BAA (HIPAA)	Sign via AWS Artifact	BAA confirmed

Development Environment

Requirement	Version	Verification
Node.js	20.x LTS	`node --version`
pnpm	8.x+	`pnpm --version`
AWS CDK CLI	2.x	`cdk --version`
Xcode	15.x+	`xcode-select -p`
Swift	5.9+	`swift --version`

AWS Service Quotas Verify these quotas before deployment (request increases if needed):

Service	Quota	Required Minimum
VPC	VPCs per region	5
Aurora	DB clusters	3 (per environment)
Lambda	Concurrent executions	1,000
API Gateway	APIs per region	10
SageMaker	Endpoint instances	20 (Tier 3+)
Secrets Manager	Secrets	100
KMS	CMKs	50

2.2 Deployment Phases

Phase 1: Bootstrap & Foundation (15-25 minutes)

```
# 1. Bootstrap CDK (one-time per account/region)
cd packages/infrastructure
```

```
cdk bootstrap aws://ACCOUNT_ID/us-east-1 --qualifier radiant

# 2. Deploy Foundation Stack
cdk deploy Radiant-dev-Foundation \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 3. Deploy Networking Stack
cdk deploy Radiant-dev-Networking \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] S3 deployment bucket created - [ ] VPC created with correct CIDR - [ ] Subnets in 3 AZs - [ ] NAT Gateway(s) active - [ ] VPC endpoints created

Phase 2: Security & Data (15-25 minutes)

```
# 4. Deploy Security Stack
cdk deploy Radiant-dev-Security \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 5. Deploy Data Stack
cdk deploy Radiant-dev-Data \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 6. Deploy Storage Stack
cdk deploy Radiant-dev-Storage \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] KMS CMK created - [ ] Secrets Manager secrets created - [ ] Aurora cluster available - [ ] DynamoDB tables created - [ ] S3 buckets created with encryption

Phase 3: Auth & AI (20-30 minutes)

```
# 7. Deploy Auth Stack
cdk deploy Radiant-dev-Auth \
  --context environment=dev \
```

```
  --context tier=1 \
  --require-approval never

# 8. Deploy AI Stack
cdk deploy Radiant-dev-AI \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] Cognito User Pool created - [ ] Cognito Admin Pool created - [ ] LiteLLM ECS service running - [ ] SageMaker endpoints active (Tier 3+)

Phase 4: API & Admin (15-20 minutes)

```
# 9. Deploy API Stack
cdk deploy Radiant-dev-API \
  --context environment=dev \
  --context tier=1 \
  --require-approval never

# 10. Deploy Admin Stack
cdk deploy Radiant-dev-Admin \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Verification: - [ ] API Gateway deployed - [ ] AppSync API created - [ ] Lambda functions deployed - [ ] CloudFront distribution active - [ ] Admin dashboard accessible

Phase 5: Database Migrations (5-10 minutes)

```
# Run migrations
cd ../../
npm run db:migrate -- --environment=dev

# Verify schema
npm run db:verify -- --environment=dev
```

Verification: - [ ] All migrations applied - [ ] RLS policies active - [ ] Indexes created - [ ] Seed data loaded

2.3 Post-Deployment Configuration

Create First Super Admin

```
# Via CLI
aws cognito-idp admin-create-user \
  --user-pool-id YOUR_ADMIN_POOL_ID \
```

```
  --username admin@YOUR_DOMAIN.com \
  --user-attributes Name=email,Value=admin@YOUR_DOMAIN.com \
  --temporary-password TempPass123! \
  --message-action SUPPRESS

aws cognito-idp admin-add-user-to-group \
  --user-pool-id YOUR_ADMIN_POOL_ID \
  --username admin@YOUR_DOMAIN.com \
  --group-name super_admin
```

Configure AI Providers

1. Navigate to Admin Dashboard Ã¢â€ â€™ Providers
2. Add API keys for each external provider:
   - OpenAI
   - Anthropic
   - Google AI
   - xAI (Grok)
   - DeepSeek
   - Others as needed
3. Verify provider connectivity with test requests

Set Pricing Configuration

1. Navigate to Admin Dashboard Ã¢â€ â€™ Billing Ã¢â€ â€™ Pricing
2. Review default markup:
   - External providers: 40%
   - Self-hosted models: 75%
3. Adjust per-model pricing if needed
4. Configure billing thresholds and alerts

PART 3: TESTING PROCEDURES

3.1 Smoke Tests

Run immediately after deployment:

```
# API Health Check
curl https://YOUR_API_ENDPOINT/health
# Expected: {"status":"healthy","version":"2.2.0"}

# GraphQL Introspection
curl -X POST https://YOUR_GRAPHQL_ENDPOINT/graphql \
  -H "Content-Type: application/json" \
  -d '{"query":"{ __schema { types { name } } }"}'
```

```
# LiteLLM Health
curl https://YOUR_LITELLM_ENDPOINT/health
# Expected: {"status":"healthy"}

# Admin Dashboard
curl -I https://YOUR_ADMIN_DOMAIN
# Expected: HTTP/2 200
```

3.2 Integration Tests

```
# Run full integration test suite
npm run test:integration -- --environment=dev

# Individual test suites
npm run test:auth       # Authentication flows
npm run test:models     # Model registry CRUD
npm run test:chat       # Chat completions
npm run test:billing    # Metering and billing
npm run test:admin      # Admin workflows
```

3.3 End-to-End Test Scenarios

Scenario 1: User Registration Ã¢â€ â€™ Chat Completion

1. Register new user via Cognito
2. Verify email and set MFA
3. Login and get access token
4. Create chat session
5. Send message with model selection
6. Verify response received
7. Verify usage metered

Scenario 2: Admin Invitation Ã¢â€ â€™ Two-Person Approval

1. Super admin creates invitation
2. Invitee receives email
3. Invitee accepts and creates account
4. Invitee logs in to admin dashboard
5. Invitee requests production action
6. First admin approves
7. Second admin approves
8. Action executes

Scenario 3: Self-Hosted Model Warm-Up (Tier 3+)

1. Verify model in COLD state
2. Send warm-up request

3. Verify transition to WARM state
4. Monitor endpoint scaling
5. Verify inference works
6. Test auto-cooling after inactivity

3.4 Compliance Verification

HIPAA Technical Safeguards

Control	Test	Pass Criteria
Access Control	Verify MFA required	All production users have MFA
Audit Controls	Check CloudTrail	All API calls logged
Integrity	Verify log integrity	File validation enabled
Transmission	Test TLS	TLS 1.3 only
Encryption	Check KMS	Per-tenant CMKs active
PHI Handling	Test sanitization	PHI redacted correctly

SOC 2 Controls

Criterion	Test	Pass Criteria
CC1	Policy review	Policies documented
CC2	Communication	Alert channels configured
CC3	Risk assessment	Findings documented
CC4	Monitoring	Dashboards active
CC5	Control activities	Approvals working
CC6	Logical access	RLS verified
CC7	System operations	Health checks passing
CC8	Change management	CI/CD with approvals
CC9	Risk mitigation	Backups verified

PART 4: SUCCESS CRITERIA

4.1 Deployment Success Criteria

All of the following must be TRUE for a successful deployment:

Infrastructure

- ☐ All CDK stacks deployed without errors
- ☐ VPC with correct CIDR and subnets
- ☐ Aurora cluster status: available
- ☐ All DynamoDB tables active
- ☐ S3 buckets with encryption enabled
- ☐ KMS CMKs with rotation enabled

Authentication

- ☐ Cognito User Pool created
- ☐ Cognito Admin Pool with MFA required (production)
- ☐ At least one super admin created
- ☐ Admin can log in successfully

AI Services

- ☐ LiteLLM ECS service: RUNNING
- ☐ At least one external provider configured
- ☐ Test completion returns valid response
- ☐ (Tier 3+) SageMaker endpoints: InService

API Layer

- ☐ API Gateway: deployed
- ☐ AppSync API: active
- ☐ /health returns 200
- ☐ GraphQL introspection works

Admin Dashboard

- ☐ CloudFront distribution: Deployed
- ☐ Dashboard loads without errors
- ☐ Navigation works
- ☐ API calls succeed

Database

- ☐ All migrations applied
- ☐ RLS policies active
- ☐ Test queries succeed
- ☐ No orphaned data

Monitoring

- ☐ CloudWatch alarms configured
- ☐ CloudTrail logging active
- ☐ GuardDuty enabled
- ☐ Security Hub enabled

4.2 Performance Benchmarks

Metric	Target	Acceptable
API Gateway latency (p50)	< 50ms	< 100ms
API Gateway latency (p99)	< 200ms	< 500ms

Metric	Target	Acceptable
Chat completion (streaming start)	< 500ms	< 1s
Admin dashboard load	< 2s	< 3s
Model warm-up time	< 3 min	< 5 min
Aurora query latency	< 10ms	< 50ms

PART 5: TROUBLESHOOTING GUIDE

5.1 Common Issues

CDK Deployment Failures

Issue	Likely Cause	Solution
Bootstrap failed	Wrong account/region	Verify `aws sts get-caller-identity`
Stack timeout	Slow resource creation	Check CloudFormation events
Resource limit	Service quota exceeded	Request quota increase
IAM permission denied	Insufficient permissions	Verify AdministratorAccess
Circular dependency	Stack references	Check stack dependencies

Aurora Connection Issues

Issue	Likely Cause	Solution
Connection refused	Security group	Verify Lambda SG can reach Aurora SG
Authentication failed	Wrong credentials	Check Secrets Manager
Timeout	VPC endpoints missing	Add RDS VPC endpoint
Too many connections	Connection pooling	Use RDS Proxy

Lambda Errors

Issue	Likely Cause	Solution
Cold start > 10s	VPC configuration	Use provisioned concurrency
Timeout	Slow downstream	Increase timeout, check dependencies

Issue	Likely Cause	Solution
Out of memory	Large payloads	Increase memory allocation
Permission denied	IAM role	Check Lambda execution role

LiteLLM Issues

Issue	Likely Cause	Solution
503 Service Unavailable	ECS not healthy	Check ECS service events
Provider timeout	API key invalid	Verify provider secrets
Rate limited	Too many requests	Implement retry with backoff
Wrong model response	Model misconfigured	Check litellm config.yaml

SageMaker Issues (Tier 3+)

Issue	Likely Cause	Solution
Endpoint failed	Out of memory	Use larger instance type
Slow cold start	Model too large	Use warm pool or smaller model
InvocationError	Model bug	Check CloudWatch logs
Capacity error	Insufficient instances	Request quota increase

5.2 Log Locations

Component	CloudWatch Log Group
API Gateway	/aws/api-gateway/radiant-{env}-api
Lambda Functions	/aws/lambda/radiant-{env}-*
LiteLLM (ECS)	/ecs/radiant-{env}-litellm
SageMaker Endpoints	/aws/sagemaker/Endpoints/radiant-*
Aurora	/aws/rds/cluster/radiant-{env}/postgresql
CloudFront	/aws/cloudfront/radiant-{env}-admin

5.3 Health Check Endpoints

```
# Platform health
curl https://api.YOUR_DOMAIN.com/health

# LiteLLM health
```

```
curl https://api.YOUR_DOMAIN.com/v2/litellm/health

# Admin API health
curl https://admin-api.YOUR_DOMAIN.com/health

# Model registry status
curl https://api.YOUR_DOMAIN.com/v2/models/status
```

5.4 Emergency Procedures

Database Restore

```
# List available snapshots
aws rds describe-db-cluster-snapshots \
  --db-cluster-identifier radiant-prod-cluster

# Restore from snapshot
aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier radiant-prod-restored \
  --snapshot-identifier your-snapshot-id \
  --engine aurora-postgresql

# Or use point-in-time recovery
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-prod-cluster \
  --db-cluster-identifier radiant-prod-recovered \
  --restore-to-time "2024-12-20T10:00:00Z"
```

Rollback Deployment

```
# Rollback to previous CDK deployment
cdk deploy Radiant-prod-API \
  --context environment=prod \
  --context tier=3 \
  --rollback
```

Disable Problematic Model

```
-- In Aurora
UPDATE models
SET status = 'disabled',
    disabled_reason = 'Emergency disable due to errors'
WHERE model_id = 'problematic-model-id';
```

PART 6: VERSION HISTORY

v2.2.0 (December 2024) - Current

New Features: - Dynamic AI Model Registry with database-driven discovery - 21 external AI provider integrations - 30+ self-hosted models across 9 categories - 5 mid-level orchestration services - Thermal state management (OFF/COLD/WARM/HOT/AUTOMATIC) - Administrator invitation system with email notifications - Two-person approval workflow for production deployments - Configurable PHI sanitization with HIPAA compliance - Multi-region deployment (US/EU/APAC) - Comprehensive billing and metering system

Architecture: - Unified Swift macOS deployment application - Next.js 14 admin dashboard - AWS CDK infrastructure as code - 9-prompt implementation structure

Breaking Changes: - PHI configuration now requires explicit setup - Admin pool separate from user pool - New database schema with RLS

v2.1.0 (December 2024)

- Added admin dashboard web application
- Implemented two-person approval workflow
- Enhanced billing service with invoicing

v2.0.0 (December 2024)

- Major architectural refactor
- Added PHI sanitization service
- Implemented compliance testing framework
- Multi-app support from single deployment

v1.1.0 (December 2024)

- Added media handling (images, video, audio, 3D)
- Implemented 20+ external provider integrations
- Enhanced metering and billing

v1.0.0 (December 2024)

- Initial release
- Basic Swift deployment app
- Core CDK infrastructure
- LiteLLM integration

DEPLOYMENT TIMELINE

Phase	Duration	Activities
Pre-deployment	1-2 hours	Account setup, quotas, certificates
Bootstrap	2-5 min	CDK bootstrap, S3 buckets
Foundation/Networking	10-15 min	VPC, subnets, NAT gateways
Security/Data	15-20 min	KMS, Secrets, Aurora, DynamoDB
Storage	5-10 min	S3 buckets, lifecycle rules
Auth	10-15 min	Cognito pools, groups
AI	15-30 min	LiteLLM, SageMaker (if Tier 3+)
API	10-15 min	API Gateway, AppSync, Lambda
Admin	10-15 min	CloudFront, dashboard sync
Migrations	5-10 min	Schema, RLS, seed data
Configuration	15-30 min	First admin, providers, pricing
Testing	30-60 min	Smoke tests, integration tests
Total	**2-4 hours**	**Complete deployment**

ESTIMATED COSTS BY TIER

Component	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Foundation	$45	$120	$350	$1,200	$4,500
AI/API	$40	$135	$625	$2,250	$9,000
Self-Hosted	$0	$0	$500	$2,000	$8,000
Total/Month	**~$85**	**~$255**	**~$1,475**	**~$5,450**	**~$21,500**

Costs are estimates and vary based on usage, region, and actual resource consumption.

FINAL NOTES

Support Resources

- **AWS Documentation**: https://docs.aws.amazon.com
- **CDK Documentation**: https://docs.aws.amazon.com/cdk
- **LiteLLM Documentation**: https://docs.litellm.ai
- **Next.js Documentation**: https://nextjs.org/docs

Recommended Monitoring Setup

1. **CloudWatch Dashboards** - Create dashboards for each component
2. **Alarms** - Set up alarms for error rates, latency, costs
3. **X-Ray** - Enable distributed tracing
4. **Cost Explorer** - Set up daily cost reports and anomaly detection

Security Best Practices

1. Rotate AWS access keys every 90 days
2. Enable MFA on root account
3. Use separate AWS accounts for prod/staging/dev
4. Regular security audits with AWS Inspector
5. Quarterly penetration testing

End of Prompt 9: Assembly & Deployment Guide RADIANT v2.2.0 - December 2024

CONGRATULATIONS! Ã°Å¸Å½â€°

You have completed the full RADIANT v2.2.0 implementation series.

What you've built: - Production-grade multi-tenant SaaS platform - 21 external AI provider integrations - 30+ self-hosted AI models - HIPAA and SOC 2 compliant infrastructure - Multi-region global deployment - Complete admin management system - Comprehensive billing and metering

Next Steps: 1. Deploy to staging and run full test suite 2. Configure all production providers 3. Complete compliance verification 4. Deploy to production with two-person approval 5. Monitor and iterate

Thank you for building with RADIANT!

â•

END OF SECTION 9

â•

â•

APPENDIX: IMPLEMENTATION VERIFICATION CHECKLIST

â•

Pre-Implementation

☐ Node.js 20+ installed (`node --version`)
☐ pnpm 8+ installed (`pnpm --version`)

- ☐ AWS CLI configured (`aws sts get-caller-identity`)
- ☐ Xcode 15+ installed (for Swift app)
- ☐ AWS CDK CLI installed (`cdk --version`)

Section 0: Shared Types

- ☐ `packages/shared/` directory created
- ☐ All type files implemented
- ☐ All constant files implemented
- ☐ All utility files implemented
- ☐ `pnpm build` succeeds in packages/shared
- ☐ `dist/` directory contains .js and .d.ts files

Section 1: Foundation & Swift App

- ☐ Root package.json configured
- ☐ Swift app compiles in Xcode
- ☐ Credentials can be saved/loaded
- ☐ Bundle scripts execute successfully

Section 2: CDK Infrastructure

- ☐ Infrastructure package imports from @radiant/shared
- ☐ NO duplicate tier/region configs created
- ☐ `cdk synth` succeeds for all stacks
- ☐ Foundation stack deploys
- ☐ Networking stack deploys
- ☐ Security stack deploys
- ☐ Data stack deploys
- ☐ Storage stack deploys

Section 3: CDK AI & API

- ☐ Auth stack deploys
- ☐ AI stack deploys
- ☐ API stack deploys
- ☐ Admin stack deploys
- ☐ Cognito user pool created
- ☐ API Gateway accessible

Section 4: Lambda Core

- ☐ All core Lambda functions implemented
- ☐ Functions import types from @radiant/shared
- ☐ Health endpoint returns 200

Section 5: Lambda Admin & Billing

- ☐ Admin Lambda functions implemented
- ☐ Uses canonical table names (administrators, invitations)
- ☐ Billing Lambda functions implemented

Section 6: Self-Hosted Models

- ☐ SageMaker endpoint configs created
- ☐ Thermal state management working
- ☐ Mid-level services defined

Section 7: Database

- ☐ All migrations in `migrations/` directory
- ☐ Migrations use canonical table names
- ☐ Migrations run successfully
- ☐ RLS policies applied

Section 8: Admin Dashboard

- ☐ Dashboard imports types from @radiant/shared
- ☐ `pnpm build` succeeds
- ☐ Dashboard accessible via CloudFront
- ☐ All pages render correctly

Section 9: Verification

- ☐ All stacks deployed
- ☐ Health checks pass
- ☐ First admin can be created
- ☐ API calls succeed
- ☐ Dashboard fully functional

Quick Domain Replacement

After cloning/creating the project, run:

```
# macOS
find . -type f \( -name "*.ts" -o -name "*.tsx" -o -name "*.json" -o -name "*.swift" -o -nam
  -not -path "./node_modules/*" \
  -exec sed -i '' 's/YOUR_DOMAIN\.com/zynapses.com/g' {} \;

# Linux
find . -type f \( -name "*.ts" -o -name "*.tsx" -o -name "*.json" -o -name "*.swift" -o -nam
```

```
  -not -path "./node_modules/*" \
  -exec sed -i 's/YOUR_DOMAIN\.com/zynapses.com/g' {} \;
```

Replace `YOUR_DOMAIN.com` with your actual domain.

SECTION-10-VISUAL-AI-PIPELINE

SECTION 10: VISUAL AI PIPELINE (v2.3.0)

10.1 Pipeline Overview

The Visual AI Pipeline extends RADIANT with 13 new self-hosted AI models for product photography and video post-production workflows.

New Models Added

Model	Category	Purpose
SAM 2 Large	Segmentation	Precise object/subject isolation
SAM 2 Base Plus	Segmentation	Balanced speed/quality
SAM 2 Small	Segmentation	Fast lightweight segmentation
SAM 2 Tiny	Segmentation	Edge deployment
XMem	Video	Temporal mask propagation
LaMa	Inpainting	Context-aware fill
RIFE	Interpolation	Frame rate upscaling
Real-ESRGAN 4x	Upscaling	Photo-realistic enhancement
Real-ESRGAN Anime	Upscaling	Animation optimization
GFPGAN	Face	Face restoration
CodeFormer	Face	Face enhancement
Background Matting V2	Matting	Alpha matte extraction
MODNet	Matting	Real-time portraits

10.2 Database Schema Extensions

```
-- migrations/020_visual_ai_pipeline.sql

-- Model thermal state tracking
ALTER TABLE self_hosted_models ADD COLUMN IF NOT EXISTS thermal_state VARCHAR(20) DEFAULT '
ALTER TABLE self_hosted_models ADD COLUMN IF NOT EXISTS warm_until TIMESTAMPTZ;
```

```sql
ALTER TABLE self_hosted_models ADD COLUMN IF NOT EXISTS auto_thermal_enabled BOOLEAN DEFAULT

-- Visual pipeline job tracking
CREATE TABLE visual_pipeline_jobs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    pipeline_type VARCHAR(50) NOT NULL,
    source_asset_key VARCHAR(500) NOT NULL,
    output_asset_key VARCHAR(500),
    models_used TEXT[] NOT NULL,
    parameters JSONB DEFAULT '{}',
    status VARCHAR(20) NOT NULL DEFAULT 'pending',
    progress INTEGER DEFAULT 0,
    started_at TIMESTAMPTZ,
    completed_at TIMESTAMPTZ,
    error_message TEXT,
    cost DECIMAL(10, 6),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_visual_jobs_tenant ON visual_pipeline_jobs(tenant_id);
CREATE INDEX idx_visual_jobs_status ON visual_pipeline_jobs(status);

ALTER TABLE visual_pipeline_jobs ENABLE ROW LEVEL SECURITY;
CREATE POLICY visual_jobs_isolation ON visual_pipeline_jobs USING (tenant_id = current_setti
```

10.3 Thermal State Service

```typescript
// packages/core/src/services/thermal-state-service.ts

import { Pool } from 'pg';
import { SageMakerClient, DescribeEndpointCommand, UpdateEndpointCommand } from '@aws-sdk/cl

export type ThermalState = 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';
export type ServiceState = 'RUNNING' | 'DEGRADED' | 'DISABLED' | 'OFFLINE';

interface ThermalConfig {
    warmDurationMinutes: number;
    hotThresholdRequestsPerMinute: number;
    coldThresholdIdleMinutes: number;
}

export class ThermalStateService {
    private pool: Pool;
    private sagemaker: SageMakerClient;
```

```
constructor(pool: Pool) {
    this.pool = pool;
    this.sagemaker = new SageMakerClient({});
}

async getThermalState(modelId: string): Promise<ThermalState> {
    const result = await this.pool.query(
        `SELECT thermal_state, warm_until, auto_thermal_enabled FROM self_hosted_models
        [modelId]
    );

    if (result.rows.length === 0) throw new Error('Model not found');

    const { thermal_state, warm_until, auto_thermal_enabled } = result.rows[0];

    if (auto_thermal_enabled && warm_until && new Date(warm_until) < new Date()) {
        await this.transitionToCold(modelId);
        return 'COLD';
    }

    return thermal_state;
}

async warmUp(modelId: string, durationMinutes: number = 30): Promise<void> {
    const warmUntil = new Date(Date.now() + durationMinutes * 60 * 1000);

    await this.pool.query(
        `UPDATE self_hosted_models SET thermal_state = 'WARM', warm_until = $2 WHERE id
        [modelId, warmUntil]
    );

    // Trigger SageMaker endpoint if needed
    const model = await this.getModel(modelId);
    if (model.endpoint_name) {
        await this.ensureEndpointRunning(model.endpoint_name);
    }
}

async transitionToCold(modelId: string): Promise<void> {
    await this.pool.query(
        `UPDATE self_hosted_models SET thermal_state = 'COLD', warm_until = NULL WHERE i
        [modelId]
    );
}
```

```
    private async getModel(modelId: string) {
        const result = await this.pool.query(`SELECT * FROM self_hosted_models WHERE id = $1
        return result.rows[0];
    }

    private async ensureEndpointRunning(endpointName: string): Promise<void> {
        const command = new DescribeEndpointCommand({ EndpointName: endpointName });
        const response = await this.sagemaker.send(command);

        if (response.EndpointStatus !== 'InService') {
            console.log(`Endpoint ${endpointName} status: ${response.EndpointStatus}`);
        }
    }
}
```

10.4 Visual Pipeline Handler

```
// packages/lambdas/visual-pipeline/handler.ts

import { S3Client, GetObjectCommand, PutObjectCommand } from '@aws-sdk/client-s3';
import { SageMakerRuntimeClient, InvokeEndpointCommand } from '@aws-sdk/client-sagemaker-run

interface PipelineRequest {
    tenantId: string;
    userId: string;
    pipelineType: 'segment' | 'inpaint' | 'upscale' | 'interpolate' | 'face_restore';
    sourceAssetKey: string;
    parameters: Record<string, any>;
}

export async function handler(event: PipelineRequest) {
    const s3 = new S3Client({});
    const sagemaker = new SageMakerRuntimeClient({});

    const pipelineHandlers: Record<string, (input: Buffer, params: any) => Promise<Buffer>>
        segment: async (input, params) => {
            const endpoint = params.quality === 'high' ? 'sam2-large' : 'sam2-base';
            return invokeSageMaker(sagemaker, endpoint, input);
        },
        inpaint: async (input, params) => {
            return invokeSageMaker(sagemaker, 'lama-inpaint', input, { mask: params.maskData
        },
        upscale: async (input, params) => {
            const endpoint = params.style === 'anime' ? 'realesrgan-anime' : 'realesrgan-4x'
            return invokeSageMaker(sagemaker, endpoint, input);
        },
```

```
        interpolate: async (input, params) => {
            return invokeSageMaker(sagemaker, 'rife-interpolation', input, { targetFps: para
        },
        face_restore: async (input, params) => {
            const endpoint = params.method === 'codeformer' ? 'codeformer' : 'gfpgan';
            return invokeSageMaker(sagemaker, endpoint, input);
        }
    };

    // Get source asset
    const sourceObj = await s3.send(new GetObjectCommand({
        Bucket: process.env.ASSETS_BUCKET!,
        Key: event.sourceAssetKey
    }));
    const inputBuffer = Buffer.from(await sourceObj.Body!.transformToByteArray());

    // Process through pipeline
    const handler = pipelineHandlers[event.pipelineType];
    const outputBuffer = await handler(inputBuffer, event.parameters);

    // Save result
    const outputKey = `processed/${event.tenantId}/${Date.now()}_${event.pipelineType}.png`;
    await s3.send(new PutObjectCommand({
        Bucket: process.env.ASSETS_BUCKET!,
        Key: outputKey,
        Body: outputBuffer,
        ContentType: 'image/png'
    }));

    return { outputAssetKey: outputKey };
}

async function invokeSageMaker(
    client: SageMakerRuntimeClient,
    endpoint: string,
    input: Buffer,
    extraParams?: Record<string, any>
): Promise<Buffer> {
    const payload = {
        image: input.toString('base64'),
        ...extraParams
    };

    const response = await client.send(new InvokeEndpointCommand({
        EndpointName: endpoint,
        Body: JSON.stringify(payload),
```

```
        ContentType: 'application/json'
    }));

    const result = JSON.parse(new TextDecoder().decode(response.Body));
    return Buffer.from(result.output, 'base64');
}
```

SECTION-11-RADIANT-BRAIN

SECTION 11: RADIANT BRAIN - SMART ROUTER (v2.4.0)

11.1 Brain Overview

RADIANT Brain is an intelligent request routing system that selects optimal models based on task analysis, cost constraints, latency requirements, and historical performance.

11.2 Brain Database Schema

```
-- migrations/021_radiant_brain.sql

CREATE TABLE brain_routing_rules (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id),
    name VARCHAR(100) NOT NULL,
    priority INTEGER NOT NULL DEFAULT 100,
    conditions JSONB NOT NULL,
    target_model VARCHAR(100) NOT NULL,
    fallback_models TEXT[],
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE brain_routing_history (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
```

```
    task_type VARCHAR(50),
    selected_model VARCHAR(100) NOT NULL,
    selection_reason TEXT,
    input_tokens INTEGER,
    output_tokens INTEGER,
    latency_ms INTEGER,
    cost DECIMAL(10, 6),
    success BOOLEAN,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_brain_history_tenant ON brain_routing_history(tenant_id);
CREATE INDEX idx_brain_history_model ON brain_routing_history(selected_model);

ALTER TABLE brain_routing_rules ENABLE ROW LEVEL SECURITY;
ALTER TABLE brain_routing_history ENABLE ROW LEVEL SECURITY;

CREATE POLICY brain_rules_isolation ON brain_routing_rules
    USING (tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY brain_history_isolation ON brain_routing_history
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
```

11.3 Brain Router Service

```
// packages/core/src/services/brain-router.ts

import { Pool } from 'pg';

interface RoutingContext {
    tenantId: string;
    userId: string;
    taskType: 'chat' | 'code' | 'analysis' | 'creative' | 'vision' | 'audio';
    inputTokenEstimate: number;
    maxLatencyMs?: number;
    maxCost?: number;
    preferredProvider?: string;
    requiresVision?: boolean;
    requiresAudio?: boolean;
}

interface RoutingResult {
    model: string;
    provider: string;
    reason: string;
    estimatedCost: number;
    estimatedLatencyMs: number;
```

```
    confidence: number;
}

export class BrainRouter {
    private pool: Pool;
    private modelPerformanceCache: Map<string, ModelPerformance> = new Map();

    constructor(pool: Pool) {
        this.pool = pool;
    }

    async route(context: RoutingContext): Promise<RoutingResult> {
        // 1. Check tenant-specific rules first
        const customRule = await this.checkCustomRules(context);
        if (customRule) return customRule;

        // 2. Get available models for task type
        const candidates = await this.getCandidateModels(context);

        // 3. Score each candidate
        const scored = await Promise.all(
            candidates.map(async (model) => ({
                model,
                score: await this.scoreModel(model, context)
            }))
        );

        // 4. Sort by score and return best match
        scored.sort((a, b) => b.score.total - a.score.total);
        const best = scored[0];

        return {
            model: best.model.model_id,
            provider: best.model.provider,
            reason: this.formatReason(best.score),
            estimatedCost: best.score.estimatedCost,
            estimatedLatencyMs: best.score.estimatedLatency,
            confidence: best.score.total
        };
    }

    private async checkCustomRules(context: RoutingContext): Promise<RoutingResult | null> {
        const result = await this.pool.query(`
            SELECT * FROM brain_routing_rules
            WHERE (tenant_id IS NULL OR tenant_id = $1)
            AND is_active = true
```

```
            ORDER BY priority ASC
        `, [context.tenantId]);

        for (const rule of result.rows) {
            if (this.matchesConditions(rule.conditions, context)) {
                return {
                    model: rule.target_model,
                    provider: this.getProviderForModel(rule.target_model),
                    reason: `Matched rule: ${rule.name}`,
                    estimatedCost: 0,
                    estimatedLatencyMs: 0,
                    confidence: 1.0
                };
            }
        }

        return null;
    }

    private async getCandidateModels(context: RoutingContext) {
        const capabilities: string[] = [];
        if (context.requiresVision) capabilities.push('vision');
        if (context.requiresAudio) capabilities.push('audio');

        const capabilityFilter = capabilities.length > 0
            ? `AND capabilities @> $2::jsonb`
            : '';

        const result = await this.pool.query(`
            SELECT * FROM external_models
            WHERE is_active = true
            ${capabilityFilter}
            UNION ALL
            SELECT * FROM self_hosted_models
            WHERE is_active = true
            ${capabilityFilter}
        `, capabilities.length > 0 ? [context.tenantId, JSON.stringify(capabilities)] : [cor

        return result.rows;
    }

    private async scoreModel(model: any, context: RoutingContext): Promise<ModelScore> {
        const perf = await this.getModelPerformance(model.model_id);

        const costScore = this.scoreCost(model, context);
        const latencyScore = this.scoreLatency(perf, context);
```

```
        const qualityScore = this.scoreQuality(model, context);
        const reliabilityScore = perf.successRate;

        const estimatedCost = this.estimateCost(model, context.inputTokenEstimate);
        const estimatedLatency = perf.avgLatencyMs;

        return {
            costScore,
            latencyScore,
            qualityScore,
            reliabilityScore,
            estimatedCost,
            estimatedLatency,
            total: (costScore * 0.25) + (latencyScore * 0.25) + (qualityScore * 0.35) + (rel
        };
    }

    private scoreCost(model: any, context: RoutingContext): number {
        if (!context.maxCost) return 0.5;
        const estimated = this.estimateCost(model, context.inputTokenEstimate);
        if (estimated > context.maxCost) return 0;
        return 1 - (estimated / context.maxCost);
    }

    private scoreLatency(perf: ModelPerformance, context: RoutingContext): number {
        if (!context.maxLatencyMs) return 0.5;
        if (perf.avgLatencyMs > context.maxLatencyMs) return 0;
        return 1 - (perf.avgLatencyMs / context.maxLatencyMs);
    }

    private scoreQuality(model: any, context: RoutingContext): number {
        const taskQuality: Record<string, Record<string, number>> = {
            'code': { 'claude-sonnet-4': 0.95, 'gpt-4o': 0.9, 'grok-4': 0.85 },
            'creative': { 'claude-opus-4': 0.95, 'gpt-4o': 0.85, 'gemini-2': 0.8 },
            'analysis': { 'claude-opus-4': 0.95, 'o1': 0.95, 'gemini-2': 0.85 }
        };
        return taskQuality[context.taskType]?.[model.model_id] ?? 0.7;
    }

    private estimateCost(model: any, inputTokens: number): number {
        const outputEstimate = inputTokens * 1.5;
        return (inputTokens * model.input_cost_per_1k / 1000) +
               (outputEstimate * model.output_cost_per_1k / 1000);
    }

    private async getModelPerformance(modelId: string): Promise<ModelPerformance> {
```

```
        if (this.modelPerformanceCache.has(modelId)) {
            return this.modelPerformanceCache.get(modelId)!;
        }

        const result = await this.pool.query(`
            SELECT
                AVG(latency_ms) as avg_latency,
                COUNT(CASE WHEN success THEN 1 END)::float / COUNT(*)::float as success_rate
            FROM brain_routing_history
            WHERE selected_model = $1
            AND created_at > NOW() - INTERVAL '7 days'
        `, [modelId]);

        const perf = {
            avgLatencyMs: result.rows[0]?.avg_latency ?? 1000,
            successRate: result.rows[0]?.success_rate ?? 0.9
        };

        this.modelPerformanceCache.set(modelId, perf);
        return perf;
    }

    private formatReason(score: ModelScore): string {
        const factors: string[] = [];
        if (score.costScore > 0.8) factors.push('cost-effective');
        if (score.latencyScore > 0.8) factors.push('fast');
        if (score.qualityScore > 0.8) factors.push('high-quality');
        if (score.reliabilityScore > 0.95) factors.push('reliable');
        return factors.join(', ') || 'balanced choice';
    }

    private matchesConditions(conditions: any, context: RoutingContext): boolean {
        if (conditions.taskType && conditions.taskType !== context.taskType) return false;
        if (conditions.minTokens && context.inputTokenEstimate < conditions.minTokens) retu
        if (conditions.maxTokens && context.inputTokenEstimate > conditions.maxTokens) retu
        return true;
    }

    private getProviderForModel(modelId: string): string {
        const providerMap: Record<string, string> = {
            'claude-opus-4': 'anthropic',
            'claude-sonnet-4': 'anthropic',
            'gpt-4o': 'openai',
            'o1': 'openai',
            'gemini-2': 'google',
            'grok-4': 'xai'
```

```
        };
        return providerMap[modelId] ?? 'unknown';
    }
}

interface ModelPerformance {
    avgLatencyMs: number;
    successRate: number;
}

interface ModelScore {
    costScore: number;
    latencyScore: number;
    qualityScore: number;
    reliabilityScore: number;
    estimatedCost: number;
    estimatedLatency: number;
    total: number;
}
```

SECTION-12-METRICS-ANALYTICS

SECTION 12: METRICS & ANALYTICS (v2.5.0)

12.1 Analytics Database Schema

```
-- migrations/022_metrics_analytics.sql

CREATE TABLE usage_metrics (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID REFERENCES users(id),
    metric_type VARCHAR(50) NOT NULL,
    metric_name VARCHAR(100) NOT NULL,
    metric_value DECIMAL(20, 6) NOT NULL,
    dimensions JSONB DEFAULT '{}',
    recorded_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
```

```
);

CREATE TABLE aggregated_metrics (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    period_start TIMESTAMPTZ NOT NULL,
    period_end TIMESTAMPTZ NOT NULL,
    period_type VARCHAR(20) NOT NULL,
    metric_type VARCHAR(50) NOT NULL,
    total_requests BIGINT DEFAULT 0,
    total_tokens BIGINT DEFAULT 0,
    total_cost DECIMAL(20, 6) DEFAULT 0,
    avg_latency_ms DECIMAL(10, 2),
    p95_latency_ms DECIMAL(10, 2),
    p99_latency_ms DECIMAL(10, 2),
    error_count INTEGER DEFAULT 0,
    unique_users INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_usage_metrics_tenant_time ON usage_metrics(tenant_id, recorded_at);
CREATE INDEX idx_usage_metrics_type ON usage_metrics(metric_type);
CREATE INDEX idx_aggregated_period ON aggregated_metrics(tenant_id, period_start, period_typ

ALTER TABLE usage_metrics ENABLE ROW LEVEL SECURITY;
ALTER TABLE aggregated_metrics ENABLE ROW LEVEL SECURITY;

CREATE POLICY usage_metrics_isolation ON usage_metrics USING (tenant_id = current_setting('a
CREATE POLICY aggregated_metrics_isolation ON aggregated_metrics USING (tenant_id = current_
```

12.2 Metrics Collector Service

```
// packages/core/src/services/metrics-collector.ts

import { Pool } from 'pg';
import { CloudWatchClient, PutMetricDataCommand } from '@aws-sdk/client-cloudwatch';

interface MetricEvent {
    tenantId: string;
    userId?: string;
    metricType: 'api_request' | 'token_usage' | 'model_inference' | 'billing';
    metricName: string;
    value: number;
    dimensions?: Record<string, string>;
}
```

```typescript
export class MetricsCollector {
    private pool: Pool;
    private cloudwatch: CloudWatchClient;
    private buffer: MetricEvent[] = [];
    private flushInterval: NodeJS.Timeout;

    constructor(pool: Pool) {
        this.pool = pool;
        this.cloudwatch = new CloudWatchClient({});
        this.flushInterval = setInterval(() => this.flush(), 10000);
    }

    record(event: MetricEvent): void {
        this.buffer.push(event);

        if (this.buffer.length >= 100) {
            this.flush();
        }
    }

    async flush(): Promise<void> {
        if (this.buffer.length === 0) return;

        const events = [...this.buffer];
        this.buffer = [];

        // Batch insert to PostgreSQL
        const values = events.map((e, i) =>
            `($${i*6+1}, $${i*6+2}, $${i*6+3}, $${i*6+4}, $${i*6+5}, $${i*6+6})`
        ).join(', ');

        const params = events.flatMap(e => [
            e.tenantId, e.userId, e.metricType, e.metricName, e.value, JSON.stringify(e.dime
        ]);

        await this.pool.query(`
            INSERT INTO usage_metrics (tenant_id, user_id, metric_type, metric_name, metric_
            VALUES ${values}
        `, params);

        // Send to CloudWatch
        await this.sendToCloudWatch(events);
    }

    private async sendToCloudWatch(events: MetricEvent[]): Promise<void> {
        const metricData = events.map(e => ({
```

```
                MetricName: e.metricName,
                Dimensions: [
                    { Name: 'TenantId', Value: e.tenantId },
                    { Name: 'MetricType', Value: e.metricType }
                ],
                Value: e.value,
                Timestamp: new Date(),
                Unit: this.getUnit(e.metricName)
            }));

            // CloudWatch accepts max 20 metrics per call
            for (let i = 0; i < metricData.length; i += 20) {
                await this.cloudwatch.send(new PutMetricDataCommand({
                    Namespace: 'RADIANT',
                    MetricData: metricData.slice(i, i + 20)
                }));
            }
        }

        private getUnit(metricName: string): string {
            if (metricName.includes('latency')) return 'Milliseconds';
            if (metricName.includes('cost')) return 'None';
            if (metricName.includes('tokens')) return 'Count';
            return 'Count';
        }

        async getAggregatedMetrics(
            tenantId: string,
            periodType: 'hourly' | 'daily' | 'weekly' | 'monthly',
            startDate: Date,
            endDate: Date
        ) {
            const result = await this.pool.query(`
                SELECT * FROM aggregated_metrics
                WHERE tenant_id = $1
                AND period_type = $2
                AND period_start >= $3
                AND period_end <= $4
                ORDER BY period_start
            `, [tenantId, periodType, startDate, endDate]);

            return result.rows;
        }
    }
```

SECTION-13-NEURAL-ENGINE

SECTION 13: USER NEURAL ENGINE (v3.0.0)

13.1 Neural Engine Overview

The User Neural Engine provides personalized AI experiences through learned preferences, conversation memory with embeddings, and adaptive behavior patterns.

13.2 Neural Engine Database Schema

```sql
-- migrations/023_user_neural_engine.sql

-- Enable pgvector for embeddings
CREATE EXTENSION IF NOT EXISTS vector;

CREATE TABLE user_preferences (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    preference_key VARCHAR(100) NOT NULL,
    preference_value JSONB NOT NULL,
    confidence DECIMAL(3, 2) DEFAULT 0.5,
    learned_from TEXT[],
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(tenant_id, user_id, preference_key)
);

CREATE TABLE user_memory (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    memory_type VARCHAR(50) NOT NULL,
    content TEXT NOT NULL,
    embedding vector(1536),
    importance DECIMAL(3, 2) DEFAULT 0.5,
    access_count INTEGER DEFAULT 0,
```

```
    last_accessed TIMESTAMPTZ,
    expires_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE user_behavior_patterns (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    pattern_type VARCHAR(50) NOT NULL,
    pattern_data JSONB NOT NULL,
    occurrence_count INTEGER DEFAULT 1,
    last_occurred TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_user_memory_embedding ON user_memory USING ivfflat (embedding vector_cosine
CREATE INDEX idx_user_preferences_user ON user_preferences(tenant_id, user_id);
CREATE INDEX idx_user_patterns_user ON user_behavior_patterns(tenant_id, user_id);

ALTER TABLE user_preferences ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_memory ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_behavior_patterns ENABLE ROW LEVEL SECURITY;

CREATE POLICY user_preferences_isolation ON user_preferences USING (tenant_id = current_sett
CREATE POLICY user_memory_isolation ON user_memory USING (tenant_id = current_setting('app.c
CREATE POLICY user_patterns_isolation ON user_behavior_patterns USING (tenant_id = current_s
```

13.3 Neural Engine Service

```
// packages/core/src/services/neural-engine.ts

import { Pool } from 'pg';
import { BedrockRuntimeClient, InvokeModelCommand } from '@aws-sdk/client-bedrock-runtime';

export class NeuralEngine {
    private pool: Pool;
    private bedrock: BedrockRuntimeClient;

    constructor(pool: Pool) {
        this.pool = pool;
        this.bedrock = new BedrockRuntimeClient({});
    }

    async learnFromConversation(
        tenantId: string,
```

```
    userId: string,
    messages: { role: string; content: string }[]
): Promise<void> {
    // Extract preferences
    const preferences = await this.extractPreferences(messages);
    for (const pref of preferences) {
        await this.updatePreference(tenantId, userId, pref.key, pref.value, pref.confide
    }

    // Store important memories
    const memories = await this.extractMemories(messages);
    for (const memory of memories) {
        await this.storeMemory(tenantId, userId, memory);
    }

    // Update behavior patterns
    await this.updateBehaviorPatterns(tenantId, userId, messages);
}

async getRelevantMemories(
    tenantId: string,
    userId: string,
    query: string,
    limit: number = 5
): Promise<any[]> {
    const embedding = await this.generateEmbedding(query);

    const result = await this.pool.query(`
        SELECT content, importance, memory_type,
               1 - (embedding <=> $4::vector) as similarity
        FROM user_memory
        WHERE tenant_id = $1 AND user_id = $2
        AND (expires_at IS NULL OR expires_at > NOW())
        ORDER BY embedding <=> $4::vector
        LIMIT $3
    `, [tenantId, userId, limit, `[${embedding.join(',')}]`]);

    // Update access counts
    for (const row of result.rows) {
        await this.pool.query(`
            UPDATE user_memory SET access_count = access_count + 1, last_accessed = NOW
            WHERE id = $1
        `, [row.id]);
    }

    return result.rows;
```

```
    }

    async getPreferences(tenantId: string, userId: string): Promise<Record<string, any>> {
        const result = await this.pool.query(`
            SELECT preference_key, preference_value, confidence
            FROM user_preferences
            WHERE tenant_id = $1 AND user_id = $2
            ORDER BY confidence DESC
        `, [tenantId, userId]);

        const prefs: Record<string, any> = {};
        for (const row of result.rows) {
            prefs[row.preference_key] = {
                value: row.preference_value,
                confidence: row.confidence
            };
        }
        return prefs;
    }

    private async generateEmbedding(text: string): Promise<number[]> {
        const response = await this.bedrock.send(new InvokeModelCommand({
            modelId: 'amazon.titan-embed-text-v1',
            body: JSON.stringify({ inputText: text }),
            contentType: 'application/json'
        }));

        const result = JSON.parse(new TextDecoder().decode(response.body));
        return result.embedding;
    }

    private async extractPreferences(messages: any[]): Promise<any[]> {
        const conversationText = messages.map(m => `${m.role}: ${m.content}`).join('\n');

        const response = await this.bedrock.send(new InvokeModelCommand({
            modelId: 'anthropic.claude-3-haiku-20240307-v1:0',
            body: JSON.stringify({
                anthropic_version: 'bedrock-2023-05-31',
                max_tokens: 1024,
                messages: [{
                    role: 'user',
                    content: `Extract user preferences from this conversation. Return JSON a

${conversationText}

Return only valid JSON array.`
```

```
            }]
        }),
        contentType: 'application/json'
    }));

    const result = JSON.parse(new TextDecoder().decode(response.body));
    try {
        return JSON.parse(result.content[0].text);
    } catch {
        return [];
    }
}

private async extractMemories(messages: any[]): Promise<any[]> {
    // Similar extraction for important facts/memories
    return [];
}

private async updatePreference(
    tenantId: string,
    userId: string,
    key: string,
    value: any,
    confidence: number
): Promise<void> {
    await this.pool.query(`
        INSERT INTO user_preferences (tenant_id, user_id, preference_key, preference_val
        VALUES ($1, $2, $3, $4, $5)
        ON CONFLICT (tenant_id, user_id, preference_key)
        DO UPDATE SET
            preference_value = EXCLUDED.preference_value,
            confidence = GREATEST(user_preferences.confidence, EXCLUDED.confidence),
            updated_at = NOW()
    `, [tenantId, userId, key, JSON.stringify(value), confidence]);
}

private async storeMemory(tenantId: string, userId: string, memory: any): Promise<void>
    const embedding = await this.generateEmbedding(memory.content);

    await this.pool.query(`
        INSERT INTO user_memory (tenant_id, user_id, memory_type, content, embedding, im
        VALUES ($1, $2, $3, $4, $5::vector, $6)
    `, [tenantId, userId, memory.type, memory.content, `[${embedding.join(',')}]`, memor
}

private async updateBehaviorPatterns(
```

```
        tenantId: string,
        userId: string,
        messages: any[]
    ): Promise<void> {
        // Pattern detection logic
    }
}
```

SECTION-14-ERROR-LOGGING

SECTION 14: CENTRALIZED ERROR LOGGING (v3.1.0)

14.1 Error Logging Database Schema

```
-- migrations/024_centralized_error_logging.sql

CREATE TABLE error_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id),
    user_id UUID REFERENCES users(id),
    error_code VARCHAR(50) NOT NULL,
    error_type VARCHAR(50) NOT NULL,
    error_message TEXT NOT NULL,
    stack_trace TEXT,
    request_id VARCHAR(100),
    source_service VARCHAR(100),
    source_function VARCHAR(200),
    context JSONB DEFAULT '{}',
    severity VARCHAR(20) NOT NULL DEFAULT 'error',
    resolved BOOLEAN DEFAULT false,
    resolved_at TIMESTAMPTZ,
    resolved_by UUID REFERENCES users(id),
    resolution_notes TEXT,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE error_patterns (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pattern_name VARCHAR(100) NOT NULL,
    error_code_pattern VARCHAR(100),
    message_pattern TEXT,
    occurrence_count INTEGER DEFAULT 1,
    first_seen TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    last_seen TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    auto_resolve_enabled BOOLEAN DEFAULT false,
    auto_resolve_action JSONB
);

CREATE INDEX idx_error_logs_tenant ON error_logs(tenant_id, created_at DESC);
CREATE INDEX idx_error_logs_code ON error_logs(error_code);
CREATE INDEX idx_error_logs_unresolved ON error_logs(resolved) WHERE resolved = false;

ALTER TABLE error_logs ENABLE ROW LEVEL SECURITY;

-- Allow admins to see all errors, users only their own
CREATE POLICY error_logs_policy ON error_logs USING (
    tenant_id = current_setting('app.current_tenant_id')::UUID OR
    current_setting('app.user_role') = 'admin'
);
```

14.2 Error Logger Service

```
// packages/core/src/services/error-logger.ts

import { Pool } from 'pg';
import { SNSClient, PublishCommand } from '@aws-sdk/client-sns';

interface ErrorContext {
    tenantId?: string;
    userId?: string;
    requestId?: string;
    sourceService: string;
    sourceFunction: string;
    additionalContext?: Record<string, any>;
}

type ErrorSeverity = 'debug' | 'info' | 'warning' | 'error' | 'critical';

export class ErrorLogger {
    private pool: Pool;
    private sns: SNSClient;
    private alertTopicArn: string;
```

```
constructor(pool: Pool, alertTopicArn: string) {
    this.pool = pool;
    this.sns = new SNSClient({});
    this.alertTopicArn = alertTopicArn;
}

async log(
    error: Error,
    context: ErrorContext,
    severity: ErrorSeverity = 'error'
): Promise<string> {
    const errorCode = this.extractErrorCode(error);
    const errorType = error.constructor.name;

    const result = await this.pool.query(`
        INSERT INTO error_logs (
            tenant_id, user_id, error_code, error_type, error_message,
            stack_trace, request_id, source_service, source_function,
            context, severity
        ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11)
        RETURNING id
    `, [
        context.tenantId,
        context.userId,
        errorCode,
        errorType,
        error.message,
        error.stack,
        context.requestId,
        context.sourceService,
        context.sourceFunction,
        JSON.stringify(context.additionalContext || {}),
        severity
    ]);

    const errorId = result.rows[0].id;

    // Check for patterns and auto-resolve
    await this.checkPatterns(errorCode, error.message);

    // Send alerts for critical errors
    if (severity === 'critical') {
        await this.sendAlert(errorId, error, context);
    }

    return errorId;
```

```
}

async getUnresolvedErrors(
    tenantId?: string,
    limit: number = 50
): Promise<any[]> {
    const whereClause = tenantId ? 'AND tenant_id = $2' : '';
    const params = tenantId ? [limit, tenantId] : [limit];

    const result = await this.pool.query(`
        SELECT * FROM error_logs
        WHERE resolved = false ${whereClause}
        ORDER BY
            CASE severity
                WHEN 'critical' THEN 1
                WHEN 'error' THEN 2
                WHEN 'warning' THEN 3
                ELSE 4
            END,
            created_at DESC
        LIMIT $1
    `, params);

    return result.rows;
}

async resolve(
    errorId: string,
    resolvedBy: string,
    notes: string
): Promise<void> {
    await this.pool.query(`
        UPDATE error_logs
        SET resolved = true, resolved_at = NOW(), resolved_by = $2, resolution_notes = $
        WHERE id = $1
    `, [errorId, resolvedBy, notes]);
}

private extractErrorCode(error: Error): string {
    if ('code' in error) return String((error as any).code);
    if (error.message.includes('ECONNREFUSED')) return 'CONN_REFUSED';
    if (error.message.includes('timeout')) return 'TIMEOUT';
    if (error.message.includes('rate limit')) return 'RATE_LIMIT';
    return 'UNKNOWN';
}
```

```
    private async checkPatterns(errorCode: string, message: string): Promise<void> {
        await this.pool.query(`
            INSERT INTO error_patterns (pattern_name, error_code_pattern, message_pattern)
            VALUES ($1, $2, $3)
            ON CONFLICT DO NOTHING
        `, [`pattern_${errorCode}`, errorCode, message.substring(0, 100)]);

        await this.pool.query(`
            UPDATE error_patterns
            SET occurrence_count = occurrence_count + 1, last_seen = NOW()
            WHERE error_code_pattern = $1
        `, [errorCode]);
    }

    private async sendAlert(errorId: string, error: Error, context: ErrorContext): Promise<v
        await this.sns.send(new PublishCommand({
            TopicArn: this.alertTopicArn,
            Subject: `[RADIANT CRITICAL] ${error.message.substring(0, 50)}`,
            Message: JSON.stringify({
                errorId,
                message: error.message,
                service: context.sourceService,
                function: context.sourceFunction,
                tenantId: context.tenantId,
                timestamp: new Date().toISOString()
            })
        }));
    }
}
```

SECTION-15-CREDENTIALS-REGISTRY

SECTION 15: EXTERNAL CREDENTIALS REGISTRY (v3.2.0)

15.1 Credentials Registry Overview

Secure storage and management of external API credentials with encryption, rotation, and access auditing.

15.2 Credentials Database Schema

```sql
-- migrations/025_credentials_registry.sql

CREATE TABLE credential_vaults (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    vault_name VARCHAR(100) NOT NULL,
    description TEXT,
    encryption_key_arn VARCHAR(500) NOT NULL,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(tenant_id, vault_name)
);

CREATE TABLE stored_credentials (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    vault_id UUID NOT NULL REFERENCES credential_vaults(id) ON DELETE CASCADE,
    credential_name VARCHAR(100) NOT NULL,
    credential_type VARCHAR(50) NOT NULL,
    encrypted_value BYTEA NOT NULL,
    metadata JSONB DEFAULT '{}',
    expires_at TIMESTAMPTZ,
    last_rotated TIMESTAMPTZ,
    rotation_schedule VARCHAR(50),
    is_active BOOLEAN DEFAULT true,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE credential_access_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    credential_id UUID NOT NULL REFERENCES stored_credentials(id),
```

```
    accessed_by UUID REFERENCES users(id),
    access_type VARCHAR(50) NOT NULL,
    access_reason TEXT,
    source_ip VARCHAR(45),
    user_agent TEXT,
    success BOOLEAN NOT NULL,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_credentials_vault ON stored_credentials(vault_id);
CREATE INDEX idx_credential_access_log ON credential_access_log(credential_id, created_at DE

ALTER TABLE credential_vaults ENABLE ROW LEVEL SECURITY;
ALTER TABLE stored_credentials ENABLE ROW LEVEL SECURITY;
ALTER TABLE credential_access_log ENABLE ROW LEVEL SECURITY;

CREATE POLICY vault_isolation ON credential_vaults USING (tenant_id = current_setting('app.c
CREATE POLICY credentials_isolation ON stored_credentials USING (
    vault_id IN (SELECT id FROM credential_vaults WHERE tenant_id = current_setting('app.cur
);
CREATE POLICY access_log_isolation ON credential_access_log USING (
    credential_id IN (
        SELECT sc.id FROM stored_credentials sc
        JOIN credential_vaults cv ON sc.vault_id = cv.id
        WHERE cv.tenant_id = current_setting('app.current_tenant_id')::UUID
    )
);
```

15.3 Credentials Manager Service

```
// packages/core/src/services/credentials-manager.ts

import { Pool } from 'pg';
import { KMSClient, EncryptCommand, DecryptCommand, GenerateDataKeyCommand } from '@aws-sdk/
import { SecretsManagerClient, GetSecretValueCommand, UpdateSecretCommand } from '@aws-sdk/c

export class CredentialsManager {
    private pool: Pool;
    private kms: KMSClient;
    private secrets: SecretsManagerClient;

    constructor(pool: Pool) {
        this.pool = pool;
        this.kms = new KMSClient({});
        this.secrets = new SecretsManagerClient({});
    }
```

```
async createVault(
    tenantId: string,
    vaultName: string,
    description?: string
): Promise<string> {
    const keyArn = await this.createKmsKey(tenantId, vaultName);

    const result = await this.pool.query(`
        INSERT INTO credential_vaults (tenant_id, vault_name, description, encryption_ke
        VALUES ($1, $2, $3, $4)
        RETURNING id
    `, [tenantId, vaultName, description, keyArn]);

    return result.rows[0].id;
}

async storeCredential(
    vaultId: string,
    name: string,
    type: string,
    value: string,
    metadata?: Record<string, any>,
    createdBy?: string
): Promise<string> {
    const vault = await this.getVault(vaultId);
    const encrypted = await this.encrypt(value, vault.encryption_key_arn);

    const result = await this.pool.query(`
        INSERT INTO stored_credentials (vault_id, credential_name, credential_type, encr
        VALUES ($1, $2, $3, $4, $5, $6)
        RETURNING id
    `, [vaultId, name, type, encrypted, JSON.stringify(metadata || {}), createdBy]);

    return result.rows[0].id;
}

async getCredential(
    credentialId: string,
    accessedBy: string,
    accessReason: string,
    sourceIp?: string
): Promise<string> {
    const cred = await this.pool.query(`
        SELECT sc.*, cv.encryption_key_arn
        FROM stored_credentials sc
```

```
            JOIN credential_vaults cv ON sc.vault_id = cv.id
            WHERE sc.id = $1 AND sc.is_active = true
        `, [credentialId]);

        if (cred.rows.length === 0) {
            await this.logAccess(credentialId, accessedBy, 'read', accessReason, sourceIp, f
            throw new Error('Credential not found or inactive');
        }

        const { encrypted_value, encryption_key_arn } = cred.rows[0];
        const decrypted = await this.decrypt(encrypted_value, encryption_key_arn);

        await this.logAccess(credentialId, accessedBy, 'read', accessReason, sourceIp, true)

        return decrypted;
    }

    async rotateCredential(
        credentialId: string,
        newValue: string,
        rotatedBy: string
    ): Promise<void> {
        const cred = await this.pool.query(`
            SELECT sc.*, cv.encryption_key_arn
            FROM stored_credentials sc
            JOIN credential_vaults cv ON sc.vault_id = cv.id
            WHERE sc.id = $1
        `, [credentialId]);

        if (cred.rows.length === 0) throw new Error('Credential not found');

        const encrypted = await this.encrypt(newValue, cred.rows[0].encryption_key_arn);

        await this.pool.query(`
            UPDATE stored_credentials
            SET encrypted_value = $2, last_rotated = NOW(), updated_at = NOW()
            WHERE id = $1
        `, [credentialId, encrypted]);

        await this.logAccess(credentialId, rotatedBy, 'rotate', 'Credential rotation', null,
    }

    private async encrypt(plaintext: string, keyArn: string): Promise<Buffer> {
        const response = await this.kms.send(new EncryptCommand({
            KeyId: keyArn,
            Plaintext: Buffer.from(plaintext)
```

```
        }));
        return Buffer.from(response.CiphertextBlob!);
    }

    private async decrypt(ciphertext: Buffer, keyArn: string): Promise<string> {
        const response = await this.kms.send(new DecryptCommand({
            KeyId: keyArn,
            CiphertextBlob: ciphertext
        }));
        return Buffer.from(response.Plaintext!).toString();
    }

    private async createKmsKey(tenantId: string, vaultName: string): Promise<string> {
        // In production, create a new KMS key
        return `arn:aws:kms:us-east-1:${process.env.AWS_ACCOUNT_ID}:key/${tenantId}-${vaultN
    }

    private async getVault(vaultId: string) {
        const result = await this.pool.query(`SELECT * FROM credential_vaults WHERE id = $1`
        if (result.rows.length === 0) throw new Error('Vault not found');
        return result.rows[0];
    }

    private async logAccess(
        credentialId: string,
        accessedBy: string,
        accessType: string,
        reason: string,
        sourceIp: string | null,
        success: boolean
    ): Promise<void> {
        await this.pool.query(`
            INSERT INTO credential_access_log (credential_id, accessed_by, access_type, acce
            VALUES ($1, $2, $3, $4, $5, $6)
        `, [credentialId, accessedBy, accessType, reason, sourceIp, success]);
    }
}
```

SECTION 16: AWS ADMIN CREDENTIALS (v3.2.0)

16.1 AWS Admin Integration

Admin-level AWS credential management for deployment operations.

```typescript
// packages/core/src/services/aws-admin-credentials.ts

import { STSClient, AssumeRoleCommand, GetCallerIdentityCommand } from '@aws-sdk/client-sts'
import { IAMClient, CreateAccessKeyCommand, DeleteAccessKeyCommand, ListAccessKeysCommand }

interface AssumedCredentials {
    accessKeyId: string;
    secretAccessKey: string;
    sessionToken: string;
    expiration: Date;
}

export class AwsAdminCredentials {
    private sts: STSClient;
    private iam: IAMClient;

    constructor() {
        this.sts = new STSClient({});
        this.iam = new IAMClient({});
    }

    async assumeDeploymentRole(
        roleArn: string,
        sessionName: string,
        durationSeconds: number = 3600
    ): Promise<AssumedCredentials> {
        const response = await this.sts.send(new AssumeRoleCommand({
            RoleArn: roleArn,
            RoleSessionName: sessionName,
            DurationSeconds: durationSeconds
        }));

        if (!response.Credentials) {
            throw new Error('Failed to assume role');
        }
```

```
            return {
                accessKeyId: response.Credentials.AccessKeyId!,
                secretAccessKey: response.Credentials.SecretAccessKey!,
                sessionToken: response.Credentials.SessionToken!,
                expiration: response.Credentials.Expiration!
            };
        }

        async validateCredentials(): Promise<{ accountId: string; arn: string }> {
            const response = await this.sts.send(new GetCallerIdentityCommand({}));
            return {
                accountId: response.Account!,
                arn: response.Arn!
            };
        }

        async rotateAccessKeys(userName: string): Promise<{ accessKeyId: string; secretAccessKey
            // List existing keys
            const listResponse = await this.iam.send(new ListAccessKeysCommand({ UserName: userN

            // Create new key
            const createResponse = await this.iam.send(new CreateAccessKeyCommand({ UserName: us

            // Delete old keys (keep only the new one)
            for (const key of listResponse.AccessKeyMetadata || []) {
                if (key.AccessKeyId !== createResponse.AccessKey?.AccessKeyId) {
                    await this.iam.send(new DeleteAccessKeyCommand({
                        UserName: userName,
                        AccessKeyId: key.AccessKeyId
                    }));
                }
            }

            return {
                accessKeyId: createResponse.AccessKey!.AccessKeyId!,
                secretAccessKey: createResponse.AccessKey!.SecretAccessKey!
            };
        }
    }
```

SECTION-17-AUTO-RESOLVE-API

SECTION 17: SIMPLE AUTO-RESOLVE API (v3.3.0)

17.1 Auto-Resolve Overview

Intelligent model selection API that automatically picks the best model based on the request.

17.2 Auto-Resolve Database Schema

```sql
-- migrations/026_auto_resolve.sql

CREATE TABLE auto_resolve_requests (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    request_type VARCHAR(50) NOT NULL,
    selected_model VARCHAR(100) NOT NULL,
    selection_reason TEXT,
    user_preferences JSONB DEFAULT '{}',
    input_tokens INTEGER,
    output_tokens INTEGER,
    cost DECIMAL(10, 6),
    latency_ms INTEGER,
    success BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_auto_resolve_tenant ON auto_resolve_requests(tenant_id, created_at DESC);
CREATE INDEX idx_auto_resolve_model ON auto_resolve_requests(selected_model);

ALTER TABLE auto_resolve_requests ENABLE ROW LEVEL SECURITY;
CREATE POLICY auto_resolve_isolation ON auto_resolve_requests USING (tenant_id = current_set
```

17.3 Auto-Resolve Service

```typescript
// packages/core/src/services/auto-resolve.ts

import { Pool } from 'pg';
import { BrainRouter } from './brain-router';
```

```typescript
interface AutoResolveRequest {
    tenantId: string;
    userId: string;
    prompt: string;
    preferences?: {
        maxCost?: number;
        maxLatencyMs?: number;
        preferredProvider?: string;
        qualityLevel?: 'economy' | 'balanced' | 'premium';
    };
}

interface AutoResolveResult {
    model: string;
    provider: string;
    reason: string;
    estimatedCost: number;
}

export class AutoResolveService {
    private pool: Pool;
    private router: BrainRouter;

    constructor(pool: Pool) {
        this.pool = pool;
        this.router = new BrainRouter(pool);
    }

    async resolve(request: AutoResolveRequest): Promise<AutoResolveResult> {
        // Analyze the prompt
        const analysis = this.analyzePrompt(request.prompt);

        // Get routing result
        const routing = await this.router.route({
            tenantId: request.tenantId,
            userId: request.userId,
            taskType: analysis.taskType,
            inputTokenEstimate: analysis.tokenEstimate,
            maxLatencyMs: request.preferences?.maxLatencyMs,
            maxCost: request.preferences?.maxCost,
            preferredProvider: request.preferences?.preferredProvider,
            requiresVision: analysis.requiresVision,
            requiresAudio: analysis.requiresAudio
        });

        // Log the request
```

```
            await this.pool.query(`
                INSERT INTO auto_resolve_requests (tenant_id, user_id, request_type, selected_mo
                VALUES ($1, $2, $3, $4, $5, $6)
            `, [
                request.tenantId,
                request.userId,
                analysis.taskType,
                routing.model,
                routing.reason,
                JSON.stringify(request.preferences || {})
            ]);

            return {
                model: routing.model,
                provider: routing.provider,
                reason: routing.reason,
                estimatedCost: routing.estimatedCost
            };
        }

        private analyzePrompt(prompt: string): {
            taskType: 'chat' | 'code' | 'analysis' | 'creative' | 'vision' | 'audio';
            tokenEstimate: number;
            requiresVision: boolean;
            requiresAudio: boolean;
        } {
            const lowerPrompt = prompt.toLowerCase();

            let taskType: 'chat' | 'code' | 'analysis' | 'creative' | 'vision' | 'audio' = 'chat

            if (lowerPrompt.includes('code') || lowerPrompt.includes('function') || lowerPrompt.
                taskType = 'code';
            } else if (lowerPrompt.includes('analyze') || lowerPrompt.includes('data') || lowerP
                taskType = 'analysis';
            } else if (lowerPrompt.includes('write') || lowerPrompt.includes('story') || lowerPr
                taskType = 'creative';
            }

            return {
                taskType,
                tokenEstimate: Math.ceil(prompt.length / 4),
                requiresVision: lowerPrompt.includes('image') || lowerPrompt.includes('picture')
                requiresAudio: lowerPrompt.includes('audio') || lowerPrompt.includes('speech')
            };
        }
    }
```

SECTION-18-THINK-TANK-PLATFORM

SECTION 18: THINK TANK CONSUMER PLATFORM (v3.5.0)

18.1 Think Tank Overview

Complex problem decomposition and multi-step reasoning with chain-of-thought processing.

18.2 Think Tank Database Schema

```
-- migrations/027_think_tank.sql

CREATE TABLE thinktank_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    problem_summary TEXT,
    domain VARCHAR(50),
    complexity VARCHAR(20),
    total_steps INTEGER DEFAULT 0,
    avg_confidence DECIMAL(3, 2),
    solution_found BOOLEAN DEFAULT false,
    total_tokens INTEGER DEFAULT 0,
    total_cost DECIMAL(10, 6) DEFAULT 0,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMPTZ
);

CREATE TABLE thinktank_steps (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES thinktank_sessions(id) ON DELETE CASCADE,
    step_number INTEGER NOT NULL,
    step_type VARCHAR(50) NOT NULL,
    description TEXT,
    reasoning TEXT,
    result TEXT,
    confidence DECIMAL(3, 2),
```

```
    model_used VARCHAR(100),
    tokens_used INTEGER,
    duration_ms INTEGER,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE thinktank_tools (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tool_name VARCHAR(100) NOT NULL UNIQUE,
    tool_type VARCHAR(50) NOT NULL,
    description TEXT,
    parameters_schema JSONB NOT NULL,
    implementation TEXT,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_thinktank_sessions_tenant ON thinktank_sessions(tenant_id, created_at DESC)
CREATE INDEX idx_thinktank_steps_session ON thinktank_steps(session_id, step_number);

ALTER TABLE thinktank_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE thinktank_steps ENABLE ROW LEVEL SECURITY;

CREATE POLICY thinktank_sessions_isolation ON thinktank_sessions USING (tenant_id = current_
CREATE POLICY thinktank_steps_isolation ON thinktank_steps USING (
    session_id IN (SELECT id FROM thinktank_sessions WHERE tenant_id = current_setting('app.
);
```

18.3 Think Tank Engine Service

```
// packages/core/src/services/thinktank-engine.ts

import { Pool } from 'pg';
import { BedrockRuntimeClient, InvokeModelCommand } from '@aws-sdk/client-bedrock-runtime';

interface ThinkTankProblem {
    tenantId: string;
    userId: string;
    problem: string;
    domain?: string;
    maxSteps?: number;
    tools?: string[];
}

interface ThinkTankStep {
    stepNumber: number;
```

```
    type: 'decompose' | 'reason' | 'execute' | 'verify' | 'synthesize';
    description: string;
    reasoning: string;
    result: string;
    confidence: number;
}

interface ThinkTankResult {
    sessionId: string;
    solution: string;
    steps: ThinkTankStep[];
    confidence: number;
    totalTokens: number;
    totalCost: number;
}

export class ThinkTankEngine {
    private pool: Pool;
    private bedrock: BedrockRuntimeClient;

    constructor(pool: Pool) {
        this.pool = pool;
        this.bedrock = new BedrockRuntimeClient({});
    }

    async solve(problem: ThinkTankProblem): Promise<ThinkTankResult> {
        // Create session
        const sessionResult = await this.pool.query(`
            INSERT INTO thinktank_sessions (tenant_id, user_id, problem_summary, domain)
            VALUES ($1, $2, $3, $4)
            RETURNING id
        `, [problem.tenantId, problem.userId, problem.problem.substring(0, 500), problem.dom

        const sessionId = sessionResult.rows[0].id;
        const steps: ThinkTankStep[] = [];
        let totalTokens = 0;
        let totalCost = 0;

        // Step 1: Decompose problem
        const decomposition = await this.decomposeProblem(problem.problem);
        steps.push(await this.recordStep(sessionId, 1, 'decompose', decomposition));
        totalTokens += decomposition.tokens;

        // Step 2-N: Solve each sub-problem
        for (let i = 0; i < decomposition.subProblems.length && i < (problem.maxSteps || 10)
            const subProblem = decomposition.subProblems[i];
```

```
        // Reason about approach
        const reasoning = await this.reason(subProblem, steps);
        steps.push(await this.recordStep(sessionId, steps.length + 1, 'reason', reasonir
        totalTokens += reasoning.tokens;

        // Execute solution
        const execution = await this.execute(subProblem, reasoning.approach, problem.too
        steps.push(await this.recordStep(sessionId, steps.length + 1, 'execute', executi
        totalTokens += execution.tokens;
    }

    // Final step: Synthesize solution
    const synthesis = await this.synthesize(problem.problem, steps);
    steps.push(await this.recordStep(sessionId, steps.length + 1, 'synthesize', synthesi
    totalTokens += synthesis.tokens;

    // Calculate cost and update session
    totalCost = totalTokens * 0.00001; // Simplified cost calculation
    const avgConfidence = steps.reduce((sum, s) => sum + s.confidence, 0) / steps.length

    await this.pool.query(`
        UPDATE thinktank_sessions
        SET total_steps = $2, avg_confidence = $3, solution_found = true,
            total_tokens = $4, total_cost = $5, completed_at = NOW()
        WHERE id = $1
    `, [sessionId, steps.length, avgConfidence, totalTokens, totalCost]);

    return {
        sessionId,
        solution: synthesis.result,
        steps,
        confidence: avgConfidence,
        totalTokens,
        totalCost
    };
}

private async decomposeProblem(problem: string): Promise<any> {
    const response = await this.invokeModel(`
        Decompose this problem into smaller sub-problems:

        Problem: ${problem}

        Return JSON: { "subProblems": ["sub1", "sub2", ...], "complexity": "low|medium|h
    `);
```

```
        return {
            subProblems: response.subProblems || [problem],
            complexity: response.complexity || 'medium',
            tokens: response.tokens,
            description: 'Problem decomposition',
            reasoning: `Identified ${response.subProblems?.length || 1} sub-problems`,
            result: JSON.stringify(response.subProblems),
            confidence: 0.9
        };
    }

    private async reason(subProblem: string, previousSteps: ThinkTankStep[]): Promise<any> {
        const context = previousSteps.map(s => s.result).join('\n');

        const response = await this.invokeModel(`
            Given context:
            ${context}

            Reason about how to solve: ${subProblem}

            Return JSON: { "approach": "description", "confidence": 0.0-1.0 }
        `);

        return {
            approach: response.approach,
            tokens: response.tokens,
            description: `Reasoning about: ${subProblem.substring(0, 50)}`,
            reasoning: response.approach,
            result: response.approach,
            confidence: response.confidence || 0.8
        };
    }

    private async execute(subProblem: string, approach: string, tools?: string[]): Promise<a
        const response = await this.invokeModel(`
            Execute this approach to solve the sub-problem:

            Sub-problem: ${subProblem}
            Approach: ${approach}
            Available tools: ${tools?.join(', ') || 'none'}

            Return JSON: { "result": "solution", "confidence": 0.0-1.0 }
        `);

        return {
```

```
            tokens: response.tokens,
            description: 'Executing solution',
            reasoning: approach,
            result: response.result,
            confidence: response.confidence || 0.8
        };
    }

    private async synthesize(originalProblem: string, steps: ThinkTankStep[]): Promise<any>
        const stepResults = steps.map(s => s.result).join('\n');

        const response = await this.invokeModel(`
            Synthesize a final solution from these steps:

            Original problem: ${originalProblem}

            Step results:
            ${stepResults}

            Return JSON: { "solution": "complete solution", "confidence": 0.0-1.0 }
        `);

        return {
            tokens: response.tokens,
            description: 'Synthesizing final solution',
            reasoning: 'Combining all step results',
            result: response.solution,
            confidence: response.confidence || 0.85
        };
    }

    private async invokeModel(prompt: string): Promise<any> {
        const response = await this.bedrock.send(new InvokeModelCommand({
            modelId: 'anthropic.claude-3-haiku-20240307-v1:0',
            body: JSON.stringify({
                anthropic_version: 'bedrock-2023-05-31',
                max_tokens: 2048,
                messages: [{ role: 'user', content: prompt }]
            }),
            contentType: 'application/json'
        }));

        const result = JSON.parse(new TextDecoder().decode(response.body));
        const text = result.content[0].text;

        try {
```

```
            const jsonMatch = text.match(/\{[\s\S]*\}/);
            const parsed = jsonMatch ? JSON.parse(jsonMatch[0]) : { result: text };
            return { ...parsed, tokens: result.usage?.output_tokens || 100 };
        } catch {
            return { result: text, tokens: result.usage?.output_tokens || 100 };
        }
    }

    private async recordStep(
        sessionId: string,
        stepNumber: number,
        stepType: string,
        data: any
    ): Promise<ThinkTankStep> {
        await this.pool.query(`
            INSERT INTO thinktank_steps (session_id, step_number, step_type, description, re
            VALUES ($1, $2, $3, $4, $5, $6, $7, $8)
        `, [sessionId, stepNumber, stepType, data.description, data.reasoning, data.result,

        return {
            stepNumber,
            type: stepType as any,
            description: data.description,
            reasoning: data.reasoning,
            result: data.result,
            confidence: data.confidence
        };
    }
}
```

SECTION-19-CONCURRENT-CHAT

SECTION 19: CONCURRENT CHAT & SPLIT-PANE UI (v3.6.0)

19.1 Concurrent Chat Overview

Industry-first feature allowing users to run multiple AI conversations simultaneously with split-pane interface.

19.2 Concurrent Chat Database Schema

```sql
-- migrations/028_concurrent_chat.sql

CREATE TABLE concurrent_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    session_name VARCHAR(100),
    layout_config JSONB NOT NULL DEFAULT '{"type": "horizontal", "panes": []}',
    max_panes INTEGER DEFAULT 4,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE concurrent_panes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES concurrent_sessions(id) ON DELETE CASCADE,
    pane_index INTEGER NOT NULL,
    chat_id UUID REFERENCES chats(id),
    model VARCHAR(100),
    status VARCHAR(20) DEFAULT 'idle',
    last_activity TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(session_id, pane_index)
);

CREATE TABLE concurrent_sync_state (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES concurrent_sessions(id) ON DELETE CASCADE,
    sync_mode VARCHAR(20) NOT NULL DEFAULT 'independent',
    shared_context TEXT,
    sync_cursor INTEGER DEFAULT 0,
```

```
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_concurrent_sessions_user ON concurrent_sessions(tenant_id, user_id);
CREATE INDEX idx_concurrent_panes_session ON concurrent_panes(session_id);

ALTER TABLE concurrent_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE concurrent_panes ENABLE ROW LEVEL SECURITY;
ALTER TABLE concurrent_sync_state ENABLE ROW LEVEL SECURITY;

CREATE POLICY concurrent_sessions_isolation ON concurrent_sessions USING (tenant_id = curren
CREATE POLICY concurrent_panes_isolation ON concurrent_panes USING (
    session_id IN (SELECT id FROM concurrent_sessions WHERE tenant_id = current_setting('app
);
CREATE POLICY concurrent_sync_isolation ON concurrent_sync_state USING (
    session_id IN (SELECT id FROM concurrent_sessions WHERE tenant_id = current_setting('app
);
```

19.3 Concurrent Session Manager

```typescript
// packages/core/src/services/concurrent-session-manager.ts

import { Pool } from 'pg';

interface LayoutConfig {
    type: 'horizontal' | 'vertical' | 'grid';
    panes: PaneConfig[];
}

interface PaneConfig {
    id: string;
    size: number;
    model?: string;
    chatId?: string;
}

export class ConcurrentSessionManager {
    private pool: Pool;

    constructor(pool: Pool) {
        this.pool = pool;
    }

    async createSession(
        tenantId: string,
        userId: string,
```

```
    name?: string,
    initialPanes: number = 2
): Promise<string> {
    const layout: LayoutConfig = {
        type: 'horizontal',
        panes: Array(initialPanes).fill(null).map((_, i) => ({
            id: `pane-${i}`,
            size: 100 / initialPanes
        }))
    };

    const result = await this.pool.query(`
        INSERT INTO concurrent_sessions (tenant_id, user_id, session_name, layout_config
        VALUES ($1, $2, $3, $4)
        RETURNING id
    `, [tenantId, userId, name, JSON.stringify(layout)]);

    const sessionId = result.rows[0].id;

    // Create pane records
    for (let i = 0; i < initialPanes; i++) {
        await this.pool.query(`
            INSERT INTO concurrent_panes (session_id, pane_index)
            VALUES ($1, $2)
        `, [sessionId, i]);
    }

    // Create sync state
    await this.pool.query(`
        INSERT INTO concurrent_sync_state (session_id)
        VALUES ($1)
    `, [sessionId]);

    return sessionId;
}

async addPane(sessionId: string, model?: string): Promise<number> {
    const session = await this.getSession(sessionId);
    if (session.layout_config.panes.length >= session.max_panes) {
        throw new Error('Maximum panes reached');
    }

    const newIndex = session.layout_config.panes.length;

    await this.pool.query(`
        INSERT INTO concurrent_panes (session_id, pane_index, model)
```

```
            VALUES ($1, $2, $3)
        `, [sessionId, newIndex, model]);

        // Update layout
        const newLayout = {
            ...session.layout_config,
            panes: [...session.layout_config.panes, { id: `pane-${newIndex}`, size: 100 / (n
        };

        // Rebalance sizes
        const equalSize = 100 / newLayout.panes.length;
        newLayout.panes = newLayout.panes.map(p => ({ ...p, size: equalSize }));

        await this.pool.query(`
            UPDATE concurrent_sessions SET layout_config = $2, updated_at = NOW() WHERE id =
        `, [sessionId, JSON.stringify(newLayout)]);

        return newIndex;
    }

    async removePane(sessionId: string, paneIndex: number): Promise<void> {
        await this.pool.query(`DELETE FROM concurrent_panes WHERE session_id = $1 AND pane_i

        // Reindex remaining panes
        await this.pool.query(`
            UPDATE concurrent_panes
            SET pane_index = pane_index - 1
            WHERE session_id = $1 AND pane_index > $2
        `, [sessionId, paneIndex]);

        // Update layout
        const session = await this.getSession(sessionId);
        const newPanes = session.layout_config.panes.filter((_, i) => i !== paneIndex);
        const equalSize = 100 / newPanes.length;

        await this.pool.query(`
            UPDATE concurrent_sessions
            SET layout_config = $2, updated_at = NOW()
            WHERE id = $1
        `, [sessionId, JSON.stringify({ ...session.layout_config, panes: newPanes.map(p => (
    }

    async updatePaneModel(sessionId: string, paneIndex: number, model: string): Promise<voi
        await this.pool.query(`
            UPDATE concurrent_panes SET model = $3 WHERE session_id = $1 AND pane_index = $2
        `, [sessionId, paneIndex, model]);
```

```
    }

    async setSyncMode(sessionId: string, mode: 'independent' | 'synchronized' | 'broadcast')
        await this.pool.query(`
            UPDATE concurrent_sync_state SET sync_mode = $2, updated_at = NOW() WHERE sessio
        `, [sessionId, mode]);
    }

    async getSession(sessionId: string) {
        const result = await this.pool.query(`SELECT * FROM concurrent_sessions WHERE id = $
        return result.rows[0];
    }

    async getPanes(sessionId: string) {
        const result = await this.pool.query(`
            SELECT * FROM concurrent_panes WHERE session_id = $1 ORDER BY pane_index
        `, [sessionId]);
        return result.rows;
    }
}
```

19.4 Split-Pane React Component

```
// apps/admin-dashboard/src/components/concurrent/SplitPane.tsx

'use client';

import React, { useState, useCallback, useRef } from 'react';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Button } from '@/components/ui/button';
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from '@/components/
import { Plus, X, ArrowLeftRight, ArrowUpDown, Grid } from 'lucide-react';

interface Pane {
    id: string;
    paneIndex: number;
    chatId?: string;
    model?: string;
    status: string;
}

interface SplitPaneProps {
    sessionId: string;
    onSendMessage: (paneIndex: number, message: string) => void;
}
```

```
export function SplitPane({ sessionId, onSendMessage }: SplitPaneProps) {
    const queryClient = useQueryClient();
    const [layout, setLayout] = useState<'horizontal' | 'vertical' | 'grid'>('horizontal');
    const [sizes, setSizes] = useState<number[]>([]);
    const containerRef = useRef<HTMLDivElement>(null);

    const { data: session } = useQuery({
        queryKey: ['concurrent-session', sessionId],
        queryFn: async () => {
            const res = await fetch(`/api/admin/concurrent/${sessionId}`);
            return res.json();
        }
    });

    const { data: panes = [] } = useQuery({
        queryKey: ['concurrent-panes', sessionId],
        queryFn: async () => {
            const res = await fetch(`/api/admin/concurrent/${sessionId}/panes`);
            return res.json();
        }
    });

    const addPaneMutation = useMutation({
        mutationFn: async () => {
            const res = await fetch(`/api/admin/concurrent/${sessionId}/panes`, { method: 'P
            return res.json();
        },
        onSuccess: () => queryClient.invalidateQueries({ queryKey: ['concurrent-panes', sess
    });

    const removePaneMutation = useMutation({
        mutationFn: async (paneIndex: number) => {
            await fetch(`/api/admin/concurrent/${sessionId}/panes/${paneIndex}`, { method: '
        },
        onSuccess: () => queryClient.invalidateQueries({ queryKey: ['concurrent-panes', sess
    });

    const updateModelMutation = useMutation({
        mutationFn: async ({ paneIndex, model }: { paneIndex: number; model: string }) => {
            await fetch(`/api/admin/concurrent/${sessionId}/panes/${paneIndex}`, {
                method: 'PATCH',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ model })
            });
        },
```

```
    onSuccess: () => queryClient.invalidateQueries({ queryKey: ['concurrent-panes', sess
});

const handleResize = useCallback((index: number, delta: number) => {
    setSizes(prev => {
        const newSizes = [...prev];
        newSizes[index] = Math.max(10, Math.min(90, newSizes[index] + delta));
        newSizes[index + 1] = Math.max(10, Math.min(90, newSizes[index + 1] - delta));
        return newSizes;
    });
}, []);

const getLayoutClasses = () => {
    switch (layout) {
        case 'vertical': return 'flex-col';
        case 'grid': return 'grid grid-cols-2';
        default: return 'flex-row';
    }
};

return (
    <div className="h-full flex flex-col">
        {/* Toolbar */}
        <div className="flex items-center justify-between p-2 border-b bg-gray-50">
            <div className="flex items-center gap-2">
                <Button
                    variant={layout === 'horizontal' ? 'default' : 'outline'}
                    size="sm"
                    onClick={() => setLayout('horizontal')}
                >
                    <ArrowLeftRight className="h-4 w-4" />
                </Button>
                <Button
                    variant={layout === 'vertical' ? 'default' : 'outline'}
                    size="sm"
                    onClick={() => setLayout('vertical')}
                >
                    <ArrowUpDown className="h-4 w-4" />
                </Button>
                <Button
                    variant={layout === 'grid' ? 'default' : 'outline'}
                    size="sm"
                    onClick={() => setLayout('grid')}
                >
                    <Grid className="h-4 w-4" />
                </Button>
```

```
    </div>

    <Button
        size="sm"
        onClick={() => addPaneMutation.mutate()}
        disabled={panes.length >= 4}
    >
        <Plus className="h-4 w-4 mr-1" /> Add Pane
    </Button>
</div>

{/* Panes Container */}
<div ref={containerRef} className={`flex-1 flex ${getLayoutClasses()} gap-1 p-1
    {panes.map((pane: Pane, index: number) => (
        <React.Fragment key={pane.id}>
            <div
                className="flex-1 min-w-0 border rounded-lg overflow-hidden"
                style={{ flex: sizes[index] ? `0 0 ${sizes[index]}%` : 1 }}
            >
                <PaneContent
                    pane={pane}
                    onRemove={() => removePaneMutation.mutate(pane.paneIndex)}
                    onModelChange={(model) => updateModelMutation.mutate({ panel
                    onSendMessage={(message) => onSendMessage(pane.paneIndex, me
                    canRemove={panes.length > 1}
                />
            </div>
            {index < panes.length - 1 && layout !== 'grid' && (
                <div
                    className={`${layout === 'horizontal' ? 'w-1 cursor-col-resi
                    onMouseDown={(e) => {
                        const startPos = layout === 'horizontal' ? e.clientX : e
                        const onMouseMove = (moveEvent: MouseEvent) => {
                            const currentPos = layout === 'horizontal' ? moveEve
                            handleResize(index, (currentPos - startPos) / 5);
                        };
                        const onMouseUp = () => {
                            document.removeEventListener('mousemove', onMouseMov
                            document.removeEventListener('mouseup', onMouseUp);
                        };
                        document.addEventListener('mousemove', onMouseMove);
                        document.addEventListener('mouseup', onMouseUp);
                    }}
                />
            )}
        </React.Fragment>
```

```
                ))}
            </div>
        </div>
    );
}

interface PaneContentProps {
    pane: Pane;
    onRemove: () => void;
    onModelChange: (model: string) => void;
    onSendMessage: (message: string) => void;
    canRemove: boolean;
}

function PaneContent({ pane, onRemove, onModelChange, onSendMessage, canRemove }: PaneConten
    const [input, setInput] = useState('');

    const models = [
        { id: 'claude-opus-4', name: 'Claude Opus 4' },
        { id: 'claude-sonnet-4', name: 'Claude Sonnet 4' },
        { id: 'gpt-4o', name: 'GPT-4o' },
        { id: 'grok-4', name: 'Grok 4' },
        { id: 'gemini-2', name: 'Gemini 2' }
    ];

    return (
        <div className="h-full flex flex-col">
            {/* Pane Header */}
            <div className="flex items-center justify-between p-2 border-b bg-white">
                <Select value={pane.model || ''} onValueChange={onModelChange}>
                    <SelectTrigger className="w-40">
                        <SelectValue placeholder="Select model" />
                    </SelectTrigger>
                    <SelectContent>
                        {models.map(m => (
                            <SelectItem key={m.id} value={m.id}>{m.name}</SelectItem>
                        ))}
                    </SelectContent>
                </Select>

                {canRemove && (
                    <Button variant="ghost" size="sm" onClick={onRemove}>
                        <X className="h-4 w-4" />
                    </Button>
                )}
            </div>
```

```
            {/* Chat Area */}
            <div className="flex-1 overflow-auto p-4 bg-gray-50">
                {/* Messages would render here */}
                <div className="text-gray-500 text-center">
                    {pane.model ? `Ready to chat with ${pane.model}` : 'Select a model to st
                </div>
            </div>

            {/* Input */}
            <div className="p-2 border-t bg-white">
                <div className="flex gap-2">
                    <input
                        type="text"
                        value={input}
                        onChange={(e) => setInput(e.target.value)}
                        placeholder="Type a message..."
                        className="flex-1 px-3 py-2 border rounded-lg"
                        onKeyDown={(e) => {
                            if (e.key === 'Enter' && input.trim()) {
                                onSendMessage(input);
                                setInput('');
                            }
                        }}
                    />
                    <Button
                        onClick={() => {
                            if (input.trim()) {
                                onSendMessage(input);
                                setInput('');
                            }
                        }}
                    >
                        Send
                    </Button>
                </div>
            </div>
        </div>
    );
}
```

SECTION 20: REAL-TIME COLLABORATION (v3.6.0)

20.1 Collaboration Overview

Real-time collaborative editing using Yjs CRDT for shared workspaces.

20.2 Collaboration Database Schema

```sql
-- migrations/029_realtime_collaboration.sql

CREATE TABLE collaboration_rooms (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    room_name VARCHAR(100) NOT NULL,
    room_type VARCHAR(50) NOT NULL DEFAULT 'document',
    created_by UUID NOT NULL REFERENCES users(id),
    yjs_document BYTEA,
    is_active BOOLEAN DEFAULT true,
    max_participants INTEGER DEFAULT 10,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE room_participants (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    room_id UUID NOT NULL REFERENCES collaboration_rooms(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),
    role VARCHAR(20) NOT NULL DEFAULT 'editor',
    cursor_position JSONB,
    is_online BOOLEAN DEFAULT false,
    joined_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    last_seen TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(room_id, user_id)
);

CREATE TABLE collaboration_history (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    room_id UUID NOT NULL REFERENCES collaboration_rooms(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),
    action_type VARCHAR(50) NOT NULL,
```

```
    action_data JSONB,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_collab_rooms_tenant ON collaboration_rooms(tenant_id);
CREATE INDEX idx_room_participants ON room_participants(room_id);
CREATE INDEX idx_collab_history ON collaboration_history(room_id, created_at DESC);

ALTER TABLE collaboration_rooms ENABLE ROW LEVEL SECURITY;
ALTER TABLE room_participants ENABLE ROW LEVEL SECURITY;
ALTER TABLE collaboration_history ENABLE ROW LEVEL SECURITY;

CREATE POLICY collab_rooms_isolation ON collaboration_rooms USING (tenant_id = current_setti
CREATE POLICY room_participants_isolation ON room_participants USING (
    room_id IN (SELECT id FROM collaboration_rooms WHERE tenant_id = current_setting('app.cu
);
CREATE POLICY collab_history_isolation ON collaboration_history USING (
    room_id IN (SELECT id FROM collaboration_rooms WHERE tenant_id = current_setting('app.cu
);
```

20.3 Yjs Collaboration Provider

```
// packages/core/src/services/yjs-provider.ts

import * as Y from 'yjs';
import { WebsocketProvider } from 'y-websocket';
import { Pool } from 'pg';

export class YjsCollaborationProvider {
    private pool: Pool;
    private documents: Map<string, Y.Doc> = new Map();
    private providers: Map<string, WebsocketProvider> = new Map();

    constructor(pool: Pool) {
        this.pool = pool;
    }

    async getOrCreateRoom(
        tenantId: string,
        roomId: string,
        userId: string
    ): Promise<{ doc: Y.Doc; provider: WebsocketProvider }> {
        // Check if document already exists in memory
        if (this.documents.has(roomId)) {
            return {
                doc: this.documents.get(roomId)!,
```

```
            provider: this.providers.get(roomId)!
        };
    }

    // Load from database or create new
    const result = await this.pool.query(
        `SELECT yjs_document FROM collaboration_rooms WHERE id = $1 AND tenant_id = $2`,
        [roomId, tenantId]
    );

    const doc = new Y.Doc();

    if (result.rows[0]?.yjs_document) {
        Y.applyUpdate(doc, result.rows[0].yjs_document);
    }

    // Create WebSocket provider
    const wsUrl = process.env.YJS_WEBSOCKET_URL || 'wss://collab.radiant.ai';
    const provider = new WebsocketProvider(wsUrl, roomId, doc);

    // Set up persistence
    doc.on('update', async (update: Uint8Array) => {
        await this.persistDocument(roomId, Y.encodeStateAsUpdate(doc));
    });

    this.documents.set(roomId, doc);
    this.providers.set(roomId, provider);

    // Add user as participant
    await this.addParticipant(roomId, userId);

    return { doc, provider };
}

async addParticipant(roomId: string, userId: string): Promise<void> {
    await this.pool.query(`
        INSERT INTO room_participants (room_id, user_id, is_online)
        VALUES ($1, $2, true)
        ON CONFLICT (room_id, user_id)
        DO UPDATE SET is_online = true, last_seen = NOW()
    `, [roomId, userId]);
}

async removeParticipant(roomId: string, userId: string): Promise<void> {
    await this.pool.query(`
        UPDATE room_participants SET is_online = false, last_seen = NOW()
```

```
            WHERE room_id = $1 AND user_id = $2
        `, [roomId, userId]);
    }

    async updateCursor(roomId: string, userId: string, position: any): Promise<void> {
        await this.pool.query(`
            UPDATE room_participants SET cursor_position = $3, last_seen = NOW()
            WHERE room_id = $1 AND user_id = $2
        `, [roomId, userId, JSON.stringify(position)]);
    }

    private async persistDocument(roomId: string, update: Uint8Array): Promise<void> {
        await this.pool.query(`
            UPDATE collaboration_rooms SET yjs_document = $2, updated_at = NOW()
            WHERE id = $1
        `, [roomId, Buffer.from(update)]);
    }

    async getParticipants(roomId: string): Promise<any[]> {
        const result = await this.pool.query(`
            SELECT rp.*, u.email, u.display_name
            FROM room_participants rp
            JOIN users u ON rp.user_id = u.id
            WHERE rp.room_id = $1
        `, [roomId]);
        return result.rows;
    }

    async logAction(roomId: string, userId: string, actionType: string, data: any): Promise<
        await this.pool.query(`
            INSERT INTO collaboration_history (room_id, user_id, action_type, action_data)
            VALUES ($1, $2, $3, $4)
        `, [roomId, userId, actionType, JSON.stringify(data)]);
    }
}
```

SECTION-21-VOICE-VIDEO

SECTION 21: VOICE & VIDEO INPUT (v3.6.0)

21.1 Voice/Video Overview

Integration with Deepgram for speech-to-text and ElevenLabs for text-to-speech.

21.2 Voice Database Schema

```sql
-- migrations/030_voice_video.sql

CREATE TABLE voice_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    session_type VARCHAR(20) NOT NULL,
    input_format VARCHAR(20),
    output_format VARCHAR(20),
    language VARCHAR(10) DEFAULT 'en',
    voice_id VARCHAR(100),
    total_duration_ms INTEGER DEFAULT 0,
    total_cost DECIMAL(10, 6) DEFAULT 0,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    ended_at TIMESTAMPTZ
);

CREATE TABLE voice_transcriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES voice_sessions(id) ON DELETE CASCADE,
    audio_url VARCHAR(500),
    transcription TEXT,
    confidence DECIMAL(3, 2),
    duration_ms INTEGER,
    word_timestamps JSONB,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE voice_synthesis (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES voice_sessions(id) ON DELETE CASCADE,
    input_text TEXT NOT NULL,
    audio_url VARCHAR(500),
```

```
    voice_id VARCHAR(100) NOT NULL,
    duration_ms INTEGER,
    character_count INTEGER,
    cost DECIMAL(10, 6),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_voice_sessions_user ON voice_sessions(tenant_id, user_id);
CREATE INDEX idx_voice_transcriptions ON voice_transcriptions(session_id);

ALTER TABLE voice_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE voice_transcriptions ENABLE ROW LEVEL SECURITY;
ALTER TABLE voice_synthesis ENABLE ROW LEVEL SECURITY;

CREATE POLICY voice_sessions_isolation ON voice_sessions USING (tenant_id = current_setting(
CREATE POLICY voice_transcriptions_isolation ON voice_transcriptions USING (
    session_id IN (SELECT id FROM voice_sessions WHERE tenant_id = current_setting('app.curr
);
CREATE POLICY voice_synthesis_isolation ON voice_synthesis USING (
    session_id IN (SELECT id FROM voice_sessions WHERE tenant_id = current_setting('app.curr
);
```

21.3 Voice Service Integration

```
// packages/core/src/services/voice-service.ts

import { Pool } from 'pg';
import { S3Client, PutObjectCommand, GetObjectCommand } from '@aws-sdk/client-s3';
import { getSignedUrl } from '@aws-sdk/s3-request-presigner';

interface DeepgramResponse {
    results: {
        channels: Array<{
            alternatives: Array<{
                transcript: string;
                confidence: number;
                words: Array<{ word: string; start: number; end: number }>;
            }>;
        }>;
    };
    metadata: {
        duration: number;
    };
}

export class VoiceService {
```

```
private pool: Pool;
private s3: S3Client;
private deepgramApiKey: string;
private elevenLabsApiKey: string;

constructor(pool: Pool) {
    this.pool = pool;
    this.s3 = new S3Client({});
    this.deepgramApiKey = process.env.DEEPGRAM_API_KEY!;
    this.elevenLabsApiKey = process.env.ELEVENLABS_API_KEY!;
}

async createSession(
    tenantId: string,
    userId: string,
    sessionType: 'stt' | 'tts' | 'realtime',
    options?: { language?: string; voiceId?: string }
): Promise<string> {
    const result = await this.pool.query(`
        INSERT INTO voice_sessions (tenant_id, user_id, session_type, language, voice_id
        VALUES ($1, $2, $3, $4, $5)
        RETURNING id
    `, [tenantId, userId, sessionType, options?.language || 'en', options?.voiceId]);

    return result.rows[0].id;
}

async transcribe(
    sessionId: string,
    audioBuffer: Buffer,
    format: string = 'webm'
): Promise<{ transcription: string; confidence: number; wordTimestamps: any[] }> {
    // Upload audio to S3
    const audioKey = `voice/${sessionId}/${Date.now()}.${format}`;
    await this.s3.send(new PutObjectCommand({
        Bucket: process.env.ASSETS_BUCKET!,
        Key: audioKey,
        Body: audioBuffer,
        ContentType: `audio/${format}`
    }));

    // Call Deepgram API
    const response = await fetch('https://api.deepgram.com/v1/listen?model=nova-2&smart_
        method: 'POST',
        headers: {
            'Authorization': `Token ${this.deepgramApiKey}`,
```

```
                'Content-Type': `audio/${format}`
            },
            body: audioBuffer
        });

        const data: DeepgramResponse = await response.json();
        const result = data.results.channels[0].alternatives[0];

        // Store transcription
        const audioUrl = await this.getSignedUrl(audioKey);
        await this.pool.query(`
            INSERT INTO voice_transcriptions (session_id, audio_url, transcription, confider
            VALUES ($1, $2, $3, $4, $5, $6)
        `, [sessionId, audioUrl, result.transcript, result.confidence, data.metadata.duratio

        return {
            transcription: result.transcript,
            confidence: result.confidence,
            wordTimestamps: result.words
        };
    }

    async synthesize(
        sessionId: string,
        text: string,
        voiceId: string = 'EXAVITQu4vr4xnSDxMaL'
    ): Promise<{ audioUrl: string; durationMs: number }> {
        // Call ElevenLabs API
        const response = await fetch(`https://api.elevenlabs.io/v1/text-to-speech/${voiceId}
            method: 'POST',
            headers: {
                'Accept': 'audio/mpeg',
                'xi-api-key': this.elevenLabsApiKey,
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                text,
                model_id: 'eleven_multilingual_v2',
                voice_settings: {
                    stability: 0.5,
                    similarity_boost: 0.75
                }
            })
        });

        const audioBuffer = Buffer.from(await response.arrayBuffer());
```

```
        // Upload to S3
        const audioKey = `voice/${sessionId}/${Date.now()}_synthesis.mp3`;
        await this.s3.send(new PutObjectCommand({
            Bucket: process.env.ASSETS_BUCKET!,
            Key: audioKey,
            Body: audioBuffer,
            ContentType: 'audio/mpeg'
        }));

        const audioUrl = await this.getSignedUrl(audioKey);
        const durationMs = this.estimateDuration(text);
        const cost = text.length * 0.00003; // Approximate ElevenLabs cost

        // Store synthesis record
        await this.pool.query(`
            INSERT INTO voice_synthesis (session_id, input_text, audio_url, voice_id, durati
            VALUES ($1, $2, $3, $4, $5, $6, $7)
        `, [sessionId, text, audioUrl, voiceId, durationMs, text.length, cost]);

        return { audioUrl, durationMs };
    }

    private async getSignedUrl(key: string): Promise<string> {
        const command = new GetObjectCommand({
            Bucket: process.env.ASSETS_BUCKET!,
            Key: key
        });
        return getSignedUrl(this.s3, command, { expiresIn: 3600 });
    }

    private estimateDuration(text: string): number {
        // Rough estimate: ~150 words per minute
        const wordCount = text.split(/\s+/).length;
        return Math.round((wordCount / 150) * 60 * 1000);
    }
}
```

SECTION-22-PERSISTENT-MEMORY

SECTION 22: PERSISTENT MEMORY (v3.6.0)

22.1 Memory System Overview

Long-term memory storage using pgvector embeddings for semantic search.

22.2 Memory Database Schema

```sql
-- migrations/031_persistent_memory.sql

CREATE TABLE memory_stores (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    store_name VARCHAR(100) NOT NULL DEFAULT 'default',
    embedding_model VARCHAR(100) DEFAULT 'text-embedding-3-small',
    total_memories INTEGER DEFAULT 0,
    last_accessed TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(tenant_id, user_id, store_name)
);

CREATE TABLE memories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    store_id UUID NOT NULL REFERENCES memory_stores(id) ON DELETE CASCADE,
    content TEXT NOT NULL,
    embedding vector(1536),
    memory_type VARCHAR(50) DEFAULT 'fact',
    source VARCHAR(100),
    importance DECIMAL(3, 2) DEFAULT 0.5,
    access_count INTEGER DEFAULT 0,
    last_accessed TIMESTAMPTZ,
    expires_at TIMESTAMPTZ,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE memory_relationships (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    source_memory_id UUID NOT NULL REFERENCES memories(id) ON DELETE CASCADE,
    target_memory_id UUID NOT NULL REFERENCES memories(id) ON DELETE CASCADE,
    relationship_type VARCHAR(50) NOT NULL,
```

```
    strength DECIMAL(3, 2) DEFAULT 0.5,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(source_memory_id, target_memory_id, relationship_type)
);

CREATE INDEX idx_memories_embedding ON memories USING ivfflat (embedding vector_cosine_ops)
CREATE INDEX idx_memories_store ON memories(store_id);
CREATE INDEX idx_memory_relationships ON memory_relationships(source_memory_id);

ALTER TABLE memory_stores ENABLE ROW LEVEL SECURITY;
ALTER TABLE memories ENABLE ROW LEVEL SECURITY;
ALTER TABLE memory_relationships ENABLE ROW LEVEL SECURITY;

CREATE POLICY memory_stores_isolation ON memory_stores USING (tenant_id = current_setting('a
CREATE POLICY memories_isolation ON memories USING (
    store_id IN (SELECT id FROM memory_stores WHERE tenant_id = current_setting('app.current
);
CREATE POLICY memory_relationships_isolation ON memory_relationships USING (
    source_memory_id IN (SELECT m.id FROM memories m JOIN memory_stores ms ON m.store_id = m
);
```

22.3 Memory Service

```
// packages/core/src/services/memory-service.ts

import { Pool } from 'pg';
import { BedrockRuntimeClient, InvokeModelCommand } from '@aws-sdk/client-bedrock-runtime';

export class MemoryService {
    private pool: Pool;
    private bedrock: BedrockRuntimeClient;

    constructor(pool: Pool) {
        this.pool = pool;
        this.bedrock = new BedrockRuntimeClient({});
    }

    async getOrCreateStore(tenantId: string, userId: string, storeName: string = 'default')
        const result = await this.pool.query(`
            INSERT INTO memory_stores (tenant_id, user_id, store_name)
            VALUES ($1, $2, $3)
            ON CONFLICT (tenant_id, user_id, store_name) DO UPDATE SET last_accessed = NOW()
            RETURNING id
        `, [tenantId, userId, storeName]);

        return result.rows[0].id;
```

```
}

async addMemory(
    storeId: string,
    content: string,
    options?: {
        type?: string;
        source?: string;
        importance?: number;
        metadata?: Record<string, any>;
    }
): Promise<string> {
    const embedding = await this.generateEmbedding(content);

    const result = await this.pool.query(`
        INSERT INTO memories (store_id, content, embedding, memory_type, source, importa
        VALUES ($1, $2, $3::vector, $4, $5, $6, $7)
        RETURNING id
    `, [
        storeId,
        content,
        `[${embedding.join(',')}]`,
        options?.type || 'fact',
        options?.source,
        options?.importance || 0.5,
        JSON.stringify(options?.metadata || {})
    ]);

    // Update store count
    await this.pool.query(`
        UPDATE memory_stores SET total_memories = total_memories + 1 WHERE id = $1
    `, [storeId]);

    return result.rows[0].id;
}

async searchMemories(
    storeId: string,
    query: string,
    limit: number = 5,
    minSimilarity: number = 0.7
): Promise<any[]> {
    const embedding = await this.generateEmbedding(query);

    const result = await this.pool.query(`
        SELECT
```

```
            id, content, memory_type, source, importance, metadata,
            1 - (embedding <=> $2::vector) as similarity
        FROM memories
        WHERE store_id = $1
        AND (expires_at IS NULL OR expires_at > NOW())
        AND 1 - (embedding <=> $2::vector) >= $4
        ORDER BY embedding <=> $2::vector
        LIMIT $3
    `, [storeId, `[${embedding.join(',')}]`, limit, minSimilarity]);

    // Update access counts
    const memoryIds = result.rows.map(r => r.id);
    if (memoryIds.length > 0) {
        await this.pool.query(`
            UPDATE memories SET access_count = access_count + 1, last_accessed = NOW()
            WHERE id = ANY($1)
        `, [memoryIds]);
    }

    return result.rows;
}

async addRelationship(
    sourceId: string,
    targetId: string,
    relationshipType: string,
    strength: number = 0.5
): Promise<void> {
    await this.pool.query(`
        INSERT INTO memory_relationships (source_memory_id, target_memory_id, relationsh
        VALUES ($1, $2, $3, $4)
        ON CONFLICT (source_memory_id, target_memory_id, relationship_type)
        DO UPDATE SET strength = EXCLUDED.strength
    `, [sourceId, targetId, relationshipType, strength]);
}

async getRelatedMemories(memoryId: string, limit: number = 5): Promise<any[]> {
    const result = await this.pool.query(`
        SELECT m.*, mr.relationship_type, mr.strength
        FROM memory_relationships mr
        JOIN memories m ON mr.target_memory_id = m.id
        WHERE mr.source_memory_id = $1
        ORDER BY mr.strength DESC
        LIMIT $2
    `, [memoryId, limit]);
```

```
        return result.rows;
    }

    private async generateEmbedding(text: string): Promise<number[]> {
        const response = await this.bedrock.send(new InvokeModelCommand({
            modelId: 'amazon.titan-embed-text-v1',
            body: JSON.stringify({ inputText: text }),
            contentType: 'application/json'
        }));

        const result = JSON.parse(new TextDecoder().decode(response.body));
        return result.embedding;
    }
}
```

SECTION-23-CANVAS-ARTIFACTS

SECTION 23: CANVAS & ARTIFACTS (v3.6.0)

23.1 Canvas Overview

Rich content editing with artifacts for code, documents, and visualizations.

23.2 Canvas Database Schema

```
-- migrations/032_canvas_artifacts.sql

CREATE TABLE canvases (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    chat_id UUID REFERENCES chats(id),
    canvas_name VARCHAR(200),
    canvas_type VARCHAR(50) NOT NULL DEFAULT 'general',
    content JSONB NOT NULL DEFAULT '{}',
    version INTEGER DEFAULT 1,
    is_published BOOLEAN DEFAULT false,
    published_url VARCHAR(500),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE artifacts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    canvas_id UUID NOT NULL REFERENCES canvases(id) ON DELETE CASCADE,
    artifact_type VARCHAR(50) NOT NULL,
    title VARCHAR(200),
    content TEXT NOT NULL,
    language VARCHAR(50),
    position_x INTEGER DEFAULT 0,
    position_y INTEGER DEFAULT 0,
    width INTEGER DEFAULT 400,
    height INTEGER DEFAULT 300,
    z_index INTEGER DEFAULT 0,
    is_collapsed BOOLEAN DEFAULT false,
    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE artifact_versions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    artifact_id UUID NOT NULL REFERENCES artifacts(id) ON DELETE CASCADE,
    version INTEGER NOT NULL,
    content TEXT NOT NULL,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_canvases_user ON canvases(tenant_id, user_id);
CREATE INDEX idx_artifacts_canvas ON artifacts(canvas_id);
CREATE INDEX idx_artifact_versions ON artifact_versions(artifact_id, version DESC);

ALTER TABLE canvases ENABLE ROW LEVEL SECURITY;
ALTER TABLE artifacts ENABLE ROW LEVEL SECURITY;
ALTER TABLE artifact_versions ENABLE ROW LEVEL SECURITY;

CREATE POLICY canvases_isolation ON canvases USING (tenant_id = current_setting('app.current
CREATE POLICY artifacts_isolation ON artifacts USING (
    canvas_id IN (SELECT id FROM canvases WHERE tenant_id = current_setting('app.current_ten
);
CREATE POLICY artifact_versions_isolation ON artifact_versions USING (
    artifact_id IN (SELECT a.id FROM artifacts a JOIN canvases c ON a.canvas_id = c.id WHERE
);
```

23.3 Canvas Service

```typescript
// packages/core/src/services/canvas-service.ts

import { Pool } from 'pg';

type ArtifactType = 'code' | 'markdown' | 'mermaid' | 'html' | 'svg' | 'json' | 'table';

interface ArtifactCreate {
    type: ArtifactType;
    title?: string;
    content: string;
    language?: string;
    position?: { x: number; y: number };
    size?: { width: number; height: number };
}

export class CanvasService {
    private pool: Pool;

    constructor(pool: Pool) {
        this.pool = pool;
    }

    async createCanvas(
        tenantId: string,
        userId: string,
        options?: { name?: string; chatId?: string; type?: string }
    ): Promise<string> {
        const result = await this.pool.query(`
            INSERT INTO canvases (tenant_id, user_id, canvas_name, chat_id, canvas_type)
            VALUES ($1, $2, $3, $4, $5)
            RETURNING id
        `, [tenantId, userId, options?.name, options?.chatId, options?.type || 'general']);

        return result.rows[0].id;
    }

    async addArtifact(canvasId: string, artifact: ArtifactCreate, createdBy?: string): Promi
        const result = await this.pool.query(`
            INSERT INTO artifacts (canvas_id, artifact_type, title, content, language, posit
            VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)
            RETURNING id
        `, [
            canvasId,
            artifact.type,
```

```
            artifact.title,
            artifact.content,
            artifact.language,
            artifact.position?.x || 0,
            artifact.position?.y || 0,
            artifact.size?.width || 400,
            artifact.size?.height || 300
        ]);

        const artifactId = result.rows[0].id;

        // Create initial version
        await this.pool.query(`
            INSERT INTO artifact_versions (artifact_id, version, content, created_by)
            VALUES ($1, 1, $2, $3)
        `, [artifactId, artifact.content, createdBy]);

        return artifactId;
    }

    async updateArtifact(artifactId: string, content: string, updatedBy?: string): Promise<v
        // Get current version
        const current = await this.pool.query(
            `SELECT COALESCE(MAX(version), 0) as max_version FROM artifact_versions WHERE ar
            [artifactId]
        );
        const newVersion = current.rows[0].max_version + 1;

        // Update artifact
        await this.pool.query(`
            UPDATE artifacts SET content = $2, updated_at = NOW() WHERE id = $1
        `, [artifactId, content]);

        // Save version
        await this.pool.query(`
            INSERT INTO artifact_versions (artifact_id, version, content, created_by)
            VALUES ($1, $2, $3, $4)
        `, [artifactId, newVersion, content, updatedBy]);
    }

    async getCanvas(canvasId: string): Promise<any> {
        const canvas = await this.pool.query(`SELECT * FROM canvases WHERE id = $1`, [canvas
        const artifacts = await this.pool.query(`SELECT * FROM artifacts WHERE canvas_id = $

        return {
            ...canvas.rows[0],
```

```
            artifacts: artifacts.rows
        };
    }

    async getArtifactVersions(artifactId: string): Promise<any[]> {
        const result = await this.pool.query(`
            SELECT av.*, u.email as created_by_email
            FROM artifact_versions av
            LEFT JOIN users u ON av.created_by = u.id
            WHERE av.artifact_id = $1
            ORDER BY av.version DESC
        `, [artifactId]);

        return result.rows;
    }

    async moveArtifact(artifactId: string, x: number, y: number): Promise<void> {
        await this.pool.query(`
            UPDATE artifacts SET position_x = $2, position_y = $3, updated_at = NOW() WHERE
        `, [artifactId, x, y]);
    }

    async resizeArtifact(artifactId: string, width: number, height: number): Promise<void> {
        await this.pool.query(`
            UPDATE artifacts SET width = $2, height = $3, updated_at = NOW() WHERE id = $1
        `, [artifactId, width, height]);
    }

    async deleteArtifact(artifactId: string): Promise<void> {
        await this.pool.query(`DELETE FROM artifacts WHERE id = $1`, [artifactId]);
    }

    async publishCanvas(canvasId: string): Promise<string> {
        const publishedUrl = `https://canvas.radiant.ai/${canvasId}`;

        await this.pool.query(`
            UPDATE canvases SET is_published = true, published_url = $2, updated_at = NOW()
        `, [canvasId, publishedUrl]);

        return publishedUrl;
    }
}
```

SECTION-24-RESULT-MERGING

SECTION 24: RESULT MERGING & AI SYNTHESIS (v3.6.0)

24.1 Result Merging Overview

Combine responses from multiple AI models with intelligent synthesis.

24.2 Merge Database Schema

```sql
-- migrations/033_result_merging.sql

CREATE TABLE merge_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    prompt TEXT NOT NULL,
    merge_strategy VARCHAR(50) NOT NULL DEFAULT 'consensus',
    models_used TEXT[] NOT NULL,
    final_result TEXT,
    quality_score DECIMAL(3, 2),
    total_cost DECIMAL(10, 6),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE merge_responses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES merge_sessions(id) ON DELETE CASCADE,
    model VARCHAR(100) NOT NULL,
    response TEXT NOT NULL,
    tokens_used INTEGER,
    latency_ms INTEGER,
    contribution_weight DECIMAL(3, 2) DEFAULT 1.0,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_merge_sessions_user ON merge_sessions(tenant_id, user_id);
CREATE INDEX idx_merge_responses_session ON merge_responses(session_id);
```

```sql
ALTER TABLE merge_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE merge_responses ENABLE ROW LEVEL SECURITY;

CREATE POLICY merge_sessions_isolation ON merge_sessions USING (tenant_id = current_setting(
CREATE POLICY merge_responses_isolation ON merge_responses USING (
    session_id IN (SELECT id FROM merge_sessions WHERE tenant_id = current_setting('app.curr
);
```

24.3 Result Merger Service

```typescript
// packages/core/src/services/result-merger.ts

import { Pool } from 'pg';
import { BedrockRuntimeClient, InvokeModelCommand } from '@aws-sdk/client-bedrock-runtime';

type MergeStrategy = 'consensus' | 'best_of' | 'synthesize' | 'debate';

interface MergeResult {
    sessionId: string;
    mergedResponse: string;
    qualityScore: number;
    contributions: Array<{ model: string; weight: number }>;
}

export class ResultMerger {
    private pool: Pool;
    private bedrock: BedrockRuntimeClient;

    constructor(pool: Pool) {
        this.pool = pool;
        this.bedrock = new BedrockRuntimeClient({});
    }

    async merge(
        tenantId: string,
        userId: string,
        prompt: string,
        responses: Array<{ model: string; response: string; tokens?: number; latencyMs?: num
        strategy: MergeStrategy = 'consensus'
    ): Promise<MergeResult> {
        // Create session
        const models = responses.map(r => r.model);
        const sessionResult = await this.pool.query(`
            INSERT INTO merge_sessions (tenant_id, user_id, prompt, merge_strategy, models_u
            VALUES ($1, $2, $3, $4, $5)
```

```
            RETURNING id
        `, [tenantId, userId, prompt, strategy, models]);

        const sessionId = sessionResult.rows[0].id;

        // Store individual responses
        for (const response of responses) {
            await this.pool.query(`
                INSERT INTO merge_responses (session_id, model, response, tokens_used, later
                VALUES ($1, $2, $3, $4, $5)
            `, [sessionId, response.model, response.response, response.tokens, response.late
        }

        // Perform merge based on strategy
        const mergeResult = await this.performMerge(prompt, responses, strategy);

        // Update session with result
        await this.pool.query(`
            UPDATE merge_sessions SET final_result = $2, quality_score = $3 WHERE id = $1
        `, [sessionId, mergeResult.mergedResponse, mergeResult.qualityScore]);

        return {
            sessionId,
            ...mergeResult
        };
    }

    private async performMerge(
        prompt: string,
        responses: Array<{ model: string; response: string }>,
        strategy: MergeStrategy
    ): Promise<{ mergedResponse: string; qualityScore: number; contributions: Array<{ model:
        const strategyHandlers: Record<MergeStrategy, () => Promise<any>> = {
            consensus: () => this.consensusMerge(responses),
            best_of: () => this.bestOfMerge(prompt, responses),
            synthesize: () => this.synthesizeMerge(prompt, responses),
            debate: () => this.debateMerge(prompt, responses)
        };

        return strategyHandlers[strategy]();
    }

    private async consensusMerge(responses: Array<{ model: string; response: string }>) {
        const responsesText = responses.map((r, i) => `Response ${i + 1} (${r.model}):\n${r.

        const synthesis = await this.invokeModel(`
```

```
            Analyze these AI responses and create a consensus response that incorporates the

            ${responsesText}

            Create a unified response that represents the consensus view. Return JSON: { "re
        `);

        return {
            mergedResponse: synthesis.response,
            qualityScore: 0.85,
            contributions: synthesis.contributions || responses.map(r => ({ model: r.model,
        };
    }

    private async bestOfMerge(prompt: string, responses: Array<{ model: string; response: st
        const responsesText = responses.map((r, i) => `Response ${i + 1} (${r.model}):\n${r.

        const evaluation = await this.invokeModel(`
            Original question: ${prompt}

            Evaluate these responses and select the best one:

            ${responsesText}

            Return JSON: { "bestIndex": 0-${responses.length - 1}, "reason": "...", "score":
        `);

        const bestResponse = responses[evaluation.bestIndex || 0];

        return {
            mergedResponse: bestResponse.response,
            qualityScore: evaluation.score || 0.8,
            contributions: [{ model: bestResponse.model, weight: 1.0 }]
        };
    }

    private async synthesizeMerge(prompt: string, responses: Array<{ model: string; response
        const responsesText = responses.map((r, i) => `Response ${i + 1} (${r.model}):\n${r.

        const synthesis = await this.invokeModel(`
            Original question: ${prompt}

            Create a comprehensive synthesis that combines insights from all these responses

            ${responsesText}
```

```
            Synthesize into a single, well-structured response that incorporates unique insi
        `);

        return {
            mergedResponse: synthesis,
            qualityScore: 0.9,
            contributions: responses.map(r => ({ model: r.model, weight: 1 / responses.lengt
        };
    }

    private async debateMerge(prompt: string, responses: Array<{ model: string; response: st
        // Multi-round debate simulation
        const responsesText = responses.map((r, i) => `${r.model}:\n${r.response}`).join('\n

        const debate = await this.invokeModel(`
            Simulate a debate between these AI perspectives on: ${prompt}

            Initial positions:
            ${responsesText}

            Identify points of agreement and disagreement, then synthesize a conclusion that
        `);

        return {
            mergedResponse: debate,
            qualityScore: 0.85,
            contributions: responses.map(r => ({ model: r.model, weight: 1 / responses.lengt
        };
    }

    private async invokeModel(prompt: string): Promise<any> {
        const response = await this.bedrock.send(new InvokeModelCommand({
            modelId: 'anthropic.claude-3-sonnet-20240229-v1:0',
            body: JSON.stringify({
                anthropic_version: 'bedrock-2023-05-31',
                max_tokens: 4096,
                messages: [{ role: 'user', content: prompt }]
            }),
            contentType: 'application/json'
        }));

        const result = JSON.parse(new TextDecoder().decode(response.body));
        const text = result.content[0].text;

        try {
            const jsonMatch = text.match(/\{[\s\S]*\}/);
```

```
            return jsonMatch ? JSON.parse(jsonMatch[0]) : text;
        } catch {
            return text;
        }
    }
}
```

SECTION-25-FOCUS-MODES-PERSONAS

SECTION 25: FOCUS MODES & CUSTOM PERSONAS (v3.6.0)

25.1 Focus Modes Overview

Pre-configured AI behavior profiles and custom persona creation.

25.2 Personas Database Schema

```
-- migrations/034_focus_personas.sql

CREATE TABLE focus_modes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id),
    mode_name VARCHAR(100) NOT NULL,
    display_name VARCHAR(200) NOT NULL,
    description TEXT,
    icon VARCHAR(50),
    system_prompt TEXT NOT NULL,
    default_model VARCHAR(100),
    settings JSONB DEFAULT '{}',
    is_system BOOLEAN DEFAULT false,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE user_personas (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
```

```
    user_id UUID NOT NULL REFERENCES users(id),
    persona_name VARCHAR(100) NOT NULL,
    display_name VARCHAR(200),
    avatar_url VARCHAR(500),
    system_prompt TEXT NOT NULL,
    voice_id VARCHAR(100),
    personality_traits JSONB DEFAULT '[]',
    knowledge_domains JSONB DEFAULT '[]',
    conversation_style JSONB DEFAULT '{}',
    is_public BOOLEAN DEFAULT false,
    usage_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE persona_usage_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    persona_id UUID NOT NULL REFERENCES user_personas(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),
    chat_id UUID REFERENCES chats(id),
    tokens_used INTEGER,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_focus_modes_tenant ON focus_modes(tenant_id) WHERE tenant_id IS NOT NULL;
CREATE INDEX idx_user_personas_user ON user_personas(tenant_id, user_id);
CREATE INDEX idx_persona_usage ON persona_usage_log(persona_id, created_at DESC);

ALTER TABLE focus_modes ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_personas ENABLE ROW LEVEL SECURITY;
ALTER TABLE persona_usage_log ENABLE ROW LEVEL SECURITY;

-- System modes visible to all, tenant modes to tenant only
CREATE POLICY focus_modes_policy ON focus_modes USING (
    is_system = true OR tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant
);
CREATE POLICY user_personas_isolation ON user_personas USING (
    tenant_id = current_setting('app.current_tenant_id')::UUID AND
    (user_id = current_setting('app.user_id')::UUID OR is_public = true)
);
CREATE POLICY persona_usage_isolation ON persona_usage_log USING (
    persona_id IN (SELECT id FROM user_personas WHERE tenant_id = current_setting('app.curre
);

-- Insert default focus modes
INSERT INTO focus_modes (mode_name, display_name, description, icon, system_prompt, is_syste
```

```
('general', 'General Assistant', 'Versatile AI for any task', 'MessageSquare', 'You are a he
('code', 'Code Expert', 'Programming and development focus', 'Code', 'You are an expert soft
('writer', 'Creative Writer', 'Creative writing and content creation', 'PenTool', 'You are a
('analyst', 'Data Analyst', 'Data analysis and insights', 'BarChart', 'You are a data analys
('researcher', 'Research Assistant', 'Deep research and fact-finding', 'Search', 'You are a
```

25.3 Persona Service

```typescript
// packages/core/src/services/persona-service.ts

import { Pool } from 'pg';

interface PersonaCreate {
    name: string;
    displayName?: string;
    systemPrompt: string;
    avatarUrl?: string;
    voiceId?: string;
    traits?: string[];
    domains?: string[];
    style?: Record<string, any>;
    isPublic?: boolean;
}

export class PersonaService {
    private pool: Pool;

    constructor(pool: Pool) {
        this.pool = pool;
    }

    async getFocusModes(tenantId?: string): Promise<any[]> {
        const result = await this.pool.query(`
            SELECT * FROM focus_modes
            WHERE is_active = true AND (is_system = true OR tenant_id IS NULL OR tenant_id =
            ORDER BY is_system DESC, mode_name
        `, [tenantId]);

        return result.rows;
    }

    async createPersona(tenantId: string, userId: string, persona: PersonaCreate): Promise<s
        const result = await this.pool.query(`
            INSERT INTO user_personas (
                tenant_id, user_id, persona_name, display_name, system_prompt,
                avatar_url, voice_id, personality_traits, knowledge_domains,
```

```
            conversation_style, is_public
        )
        VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11)
        RETURNING id
    `, [
        tenantId,
        userId,
        persona.name,
        persona.displayName,
        persona.systemPrompt,
        persona.avatarUrl,
        persona.voiceId,
        JSON.stringify(persona.traits || []),
        JSON.stringify(persona.domains || []),
        JSON.stringify(persona.style || {}),
        persona.isPublic || false
    ]);

    return result.rows[0].id;
}

async getUserPersonas(tenantId: string, userId: string): Promise<any[]> {
    const result = await this.pool.query(`
        SELECT * FROM user_personas
        WHERE tenant_id = $1 AND (user_id = $2 OR is_public = true)
        ORDER BY usage_count DESC
    `, [tenantId, userId]);

    return result.rows;
}

async getPersona(personaId: string): Promise<any> {
    const result = await this.pool.query(`SELECT * FROM user_personas WHERE id = $1`, [p
    return result.rows[0];
}

async updatePersona(personaId: string, updates: Partial<PersonaCreate>): Promise<void> {
    const fields: string[] = [];
    const values: any[] = [personaId];
    let paramIndex = 2;

    if (updates.name) { fields.push(`persona_name = $${paramIndex++}`); values.push(upda
    if (updates.displayName) { fields.push(`display_name = $${paramIndex++}`); values.pu
    if (updates.systemPrompt) { fields.push(`system_prompt = $${paramIndex++}`); values
    if (updates.avatarUrl) { fields.push(`avatar_url = $${paramIndex++}`); values.push(u
    if (updates.voiceId) { fields.push(`voice_id = $${paramIndex++}`); values.push(updat
```

```
        if (updates.traits) { fields.push(`personality_traits = $${paramIndex++}`); values.p
        if (updates.domains) { fields.push(`knowledge_domains = $${paramIndex++}`); values.p
        if (updates.style) { fields.push(`conversation_style = $${paramIndex++}`); values.pu
        if (typeof updates.isPublic === 'boolean') { fields.push(`is_public = $${paramIndex+

        fields.push('updated_at = NOW()');

        await this.pool.query(`UPDATE user_personas SET ${fields.join(', ')} WHERE id = $1`,
    }

    async logUsage(personaId: string, userId: string, chatId?: string, tokensUsed?: number):
        await this.pool.query(`
            INSERT INTO persona_usage_log (persona_id, user_id, chat_id, tokens_used)
            VALUES ($1, $2, $3, $4)
        `, [personaId, userId, chatId, tokensUsed]);

        await this.pool.query(`
            UPDATE user_personas SET usage_count = usage_count + 1 WHERE id = $1
        `, [personaId]);
    }

    async buildPrompt(personaId: string, userMessage: string): Promise<string> {
        const persona = await this.getPersona(personaId);
        if (!persona) throw new Error('Persona not found');

        const traits = persona.personality_traits as string[];
        const domains = persona.knowledge_domains as string[];

        let prompt = persona.system_prompt;

        if (traits.length > 0) {
            prompt += `\n\nPersonality traits: ${traits.join(', ')}.`;
        }

        if (domains.length > 0) {
            prompt += `\n\nAreas of expertise: ${domains.join(', ')}.`;
        }

        return prompt;
    }
}
```

SECTION-26-SCHEDULED-PROMPTS

SECTION 26: SCHEDULED PROMPTS (v3.6.0)

26.1 Scheduled Prompts Overview

Schedule AI tasks to run at specific times or intervals.

26.2 Scheduled Prompts Database Schema

```sql
-- migrations/035_scheduled_prompts.sql

CREATE TABLE scheduled_prompts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    prompt_name VARCHAR(200) NOT NULL,
    prompt_text TEXT NOT NULL,
    model VARCHAR(100) NOT NULL,
    schedule_type VARCHAR(20) NOT NULL,
    cron_expression VARCHAR(100),
    run_at TIMESTAMPTZ,
    timezone VARCHAR(50) DEFAULT 'UTC',
    is_active BOOLEAN DEFAULT true,
    max_runs INTEGER,
    run_count INTEGER DEFAULT 0,
    last_run TIMESTAMPTZ,
    next_run TIMESTAMPTZ,
    notification_email VARCHAR(255),
    output_destination VARCHAR(50) DEFAULT 'email',
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE scheduled_prompt_runs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    prompt_id UUID NOT NULL REFERENCES scheduled_prompts(id) ON DELETE CASCADE,
    status VARCHAR(20) NOT NULL DEFAULT 'pending',
    output TEXT,
    tokens_used INTEGER,
    cost DECIMAL(10, 6),
    latency_ms INTEGER,
    error_message TEXT,
```

```
    started_at TIMESTAMPTZ,
    completed_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_scheduled_prompts_user ON scheduled_prompts(tenant_id, user_id);
CREATE INDEX idx_scheduled_prompts_next_run ON scheduled_prompts(next_run) WHERE is_active =
CREATE INDEX idx_prompt_runs ON scheduled_prompt_runs(prompt_id, created_at DESC);

ALTER TABLE scheduled_prompts ENABLE ROW LEVEL SECURITY;
ALTER TABLE scheduled_prompt_runs ENABLE ROW LEVEL SECURITY;

CREATE POLICY scheduled_prompts_isolation ON scheduled_prompts USING (tenant_id = current_se
CREATE POLICY prompt_runs_isolation ON scheduled_prompt_runs USING (
    prompt_id IN (SELECT id FROM scheduled_prompts WHERE tenant_id = current_setting('app.cu
);
```

26.3 Scheduler Service

```
// packages/core/src/services/scheduler-service.ts

import { Pool } from 'pg';
import { EventBridgeClient, PutRuleCommand, PutTargetsCommand, DeleteRuleCommand } from '@aw
import * as cronParser from 'cron-parser';

type ScheduleType = 'once' | 'cron' | 'interval';

interface ScheduleCreate {
    name: string;
    prompt: string;
    model: string;
    type: ScheduleType;
    cronExpression?: string;
    runAt?: Date;
    intervalMinutes?: number;
    timezone?: string;
    maxRuns?: number;
    notificationEmail?: string;
    outputDestination?: 'email' | 'webhook' | 'storage';
}

export class SchedulerService {
    private pool: Pool;
    private eventBridge: EventBridgeClient;

    constructor(pool: Pool) {
```

```
        this.pool = pool;
        this.eventBridge = new EventBridgeClient({});
    }

    async createSchedule(tenantId: string, userId: string, schedule: ScheduleCreate): Promis
        const nextRun = this.calculateNextRun(schedule);

        const result = await this.pool.query(`
            INSERT INTO scheduled_prompts (
                tenant_id, user_id, prompt_name, prompt_text, model, schedule_type,
                cron_expression, run_at, timezone, max_runs, next_run,
                notification_email, output_destination
            )
            VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13)
            RETURNING id
        `, [
            tenantId, userId, schedule.name, schedule.prompt, schedule.model, schedule.type,
            schedule.cronExpression, schedule.runAt, schedule.timezone || 'UTC', schedule.ma
            nextRun, schedule.notificationEmail, schedule.outputDestination || 'email'
        ]);

        const promptId = result.rows[0].id;

        // Create EventBridge rule for cron schedules
        if (schedule.type === 'cron' && schedule.cronExpression) {
            await this.createEventBridgeRule(promptId, schedule.cronExpression);
        }

        return promptId;
    }

    async executeScheduledPrompt(promptId: string): Promise<string> {
        const prompt = await this.getScheduledPrompt(promptId);
        if (!prompt || !prompt.is_active) {
            throw new Error('Scheduled prompt not found or inactive');
        }

        // Create run record
        const runResult = await this.pool.query(`
            INSERT INTO scheduled_prompt_runs (prompt_id, status, started_at)
            VALUES ($1, 'running', NOW())
            RETURNING id
        `, [promptId]);

        const runId = runResult.rows[0].id;
```

```
    try {
        // Execute the prompt (would call AI service)
        const startTime = Date.now();
        const output = await this.executePrompt(prompt.prompt_text, prompt.model);
        const latencyMs = Date.now() - startTime;

        // Update run record
        await this.pool.query(`
            UPDATE scheduled_prompt_runs
            SET status = 'completed', output = $2, latency_ms = $3, completed_at = NOW()
            WHERE id = $1
        `, [runId, output, latencyMs]);

        // Update schedule
        const nextRun = this.calculateNextRun({
            type: prompt.schedule_type,
            cronExpression: prompt.cron_expression
        });

        await this.pool.query(`
            UPDATE scheduled_prompts
            SET last_run = NOW(), run_count = run_count + 1, next_run = $2
            WHERE id = $1
        `, [promptId, nextRun]);

        // Check if max runs reached
        if (prompt.max_runs && prompt.run_count + 1 >= prompt.max_runs) {
            await this.pool.query(`UPDATE scheduled_prompts SET is_active = false WHERE
        }

        return runId;
    } catch (error: any) {
        await this.pool.query(`
            UPDATE scheduled_prompt_runs
            SET status = 'failed', error_message = $2, completed_at = NOW()
            WHERE id = $1
        `, [runId, error.message]);

        throw error;
    }
}

async getScheduledPrompt(promptId: string): Promise<any> {
    const result = await this.pool.query(`SELECT * FROM scheduled_prompts WHERE id = $1
    return result.rows[0];
}
```

```
async getUserSchedules(tenantId: string, userId: string): Promise<any[]> {
    const result = await this.pool.query(`
        SELECT sp.*,
               (SELECT COUNT(*) FROM scheduled_prompt_runs WHERE prompt_id = sp.id) as t
               (SELECT status FROM scheduled_prompt_runs WHERE prompt_id = sp.id ORDER B
        FROM scheduled_prompts sp
        WHERE sp.tenant_id = $1 AND sp.user_id = $2
        ORDER BY sp.created_at DESC
    `, [tenantId, userId]);

    return result.rows;
}

async pauseSchedule(promptId: string): Promise<void> {
    await this.pool.query(`UPDATE scheduled_prompts SET is_active = false WHERE id = $1`
}

async resumeSchedule(promptId: string): Promise<void> {
    const nextRun = this.calculateNextRun(await this.getScheduledPrompt(promptId));
    await this.pool.query(`
        UPDATE scheduled_prompts SET is_active = true, next_run = $2 WHERE id = $1
    `, [promptId, nextRun]);
}

private calculateNextRun(schedule: any): Date | null {
    if (schedule.type === 'once' && schedule.runAt) {
        return new Date(schedule.runAt);
    }

    if (schedule.type === 'cron' && schedule.cronExpression) {
        const interval = cronParser.parseExpression(schedule.cronExpression);
        return interval.next().toDate();
    }

    return null;
}

private async createEventBridgeRule(promptId: string, cronExpression: string): Promise<v
    const ruleName = `radiant-schedule-${promptId}`;

    await this.eventBridge.send(new PutRuleCommand({
        Name: ruleName,
        ScheduleExpression: `cron(${cronExpression})`,
        State: 'ENABLED'
    }));
```

```
        await this.eventBridge.send(new PutTargetsCommand({
            Rule: ruleName,
            Targets: [{
                Id: 'scheduled-prompt-target',
                Arn: process.env.SCHEDULER_LAMBDA_ARN!,
                Input: JSON.stringify({ promptId })
            }]
        }));
    }

    private async executePrompt(prompt: string, model: string): Promise<string> {
        // This would call the AI service
        return `Executed prompt with model ${model}`;
    }
}
```

SECTION-27-FAMILY-TEAM-PLANS

SECTION 27: FAMILY & TEAM PLANS (v3.6.0)

27.1 Team Plans Overview

Shared subscription plans for families and teams with usage allocation.

27.2 Team Plans Database Schema

```
-- migrations/036_team_plans.sql

CREATE TABLE team_plans (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    plan_name VARCHAR(100) NOT NULL,
    plan_type VARCHAR(20) NOT NULL,
    owner_id UUID NOT NULL REFERENCES users(id),
    max_members INTEGER NOT NULL DEFAULT 5,
    total_tokens_monthly BIGINT NOT NULL,
```

```
    shared_pool BOOLEAN DEFAULT true,
    billing_email VARCHAR(255),
    stripe_subscription_id VARCHAR(100),
    is_active BOOLEAN DEFAULT true,
    current_period_start TIMESTAMPTZ,
    current_period_end TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE team_members (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID NOT NULL REFERENCES team_plans(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),
    role VARCHAR(20) NOT NULL DEFAULT 'member',
    token_allocation BIGINT,
    tokens_used_this_period BIGINT DEFAULT 0,
    invited_by UUID REFERENCES users(id),
    invited_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    accepted_at TIMESTAMPTZ,
    is_active BOOLEAN DEFAULT true,
    UNIQUE(team_id, user_id)
);

CREATE TABLE team_usage_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID NOT NULL REFERENCES team_plans(id) ON DELETE CASCADE,
    member_id UUID NOT NULL REFERENCES team_members(id) ON DELETE CASCADE,
    tokens_used INTEGER NOT NULL,
    model VARCHAR(100),
    usage_type VARCHAR(50),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_team_plans_tenant ON team_plans(tenant_id);
CREATE INDEX idx_team_members_user ON team_members(user_id);
CREATE INDEX idx_team_usage ON team_usage_log(team_id, created_at DESC);

ALTER TABLE team_plans ENABLE ROW LEVEL SECURITY;
ALTER TABLE team_members ENABLE ROW LEVEL SECURITY;
ALTER TABLE team_usage_log ENABLE ROW LEVEL SECURITY;

CREATE POLICY team_plans_isolation ON team_plans USING (tenant_id = current_setting('app.cu
CREATE POLICY team_members_isolation ON team_members USING (
    team_id IN (SELECT id FROM team_plans WHERE tenant_id = current_setting('app.current_te
);
CREATE POLICY team_usage_isolation ON team_usage_log USING (
```

```
    team_id IN (SELECT id FROM team_plans WHERE tenant_id = current_setting('app.current_ten
);
```

27.3 Team Service

```typescript
// packages/core/src/services/team-service.ts

import { Pool } from 'pg';

type PlanType = 'family' | 'team' | 'enterprise';
type MemberRole = 'owner' | 'admin' | 'member';

interface TeamCreate {
    name: string;
    type: PlanType;
    maxMembers: number;
    totalTokensMonthly: number;
    sharedPool?: boolean;
    billingEmail?: string;
}

export class TeamService {
    private pool: Pool;

    constructor(pool: Pool) {
        this.pool = pool;
    }

    async createTeam(tenantId: string, ownerId: string, team: TeamCreate): Promise<string> {
        const result = await this.pool.query(`
            INSERT INTO team_plans (
                tenant_id, plan_name, plan_type, owner_id, max_members,
                total_tokens_monthly, shared_pool, billing_email
            )
            VALUES ($1, $2, $3, $4, $5, $6, $7, $8)
            RETURNING id
        `, [
            tenantId, team.name, team.type, ownerId, team.maxMembers,
            team.totalTokensMonthly, team.sharedPool ?? true, team.billingEmail
        ]);

        const teamId = result.rows[0].id;

        // Add owner as first member
        await this.addMember(teamId, ownerId, 'owner');
```

```
    return teamId;
}

async addMember(teamId: string, userId: string, role: MemberRole = 'member', invitedBy?:
    const team = await this.getTeam(teamId);
    const memberCount = await this.getMemberCount(teamId);

    if (memberCount >= team.max_members) {
        throw new Error('Team member limit reached');
    }

    const result = await this.pool.query(`
        INSERT INTO team_members (team_id, user_id, role, invited_by, accepted_at)
        VALUES ($1, $2, $3, $4, CASE WHEN $3 = 'owner' THEN NOW() ELSE NULL END)
        RETURNING id
    `, [teamId, userId, role, invitedBy]);

    return result.rows[0].id;
}

async removeMember(teamId: string, userId: string): Promise<void> {
    // Check not removing owner
    const member = await this.getMember(teamId, userId);
    if (member?.role === 'owner') {
        throw new Error('Cannot remove team owner');
    }

    await this.pool.query(`
        UPDATE team_members SET is_active = false WHERE team_id = $1 AND user_id = $2
    `, [teamId, userId]);
}

async acceptInvitation(teamId: string, userId: string): Promise<void> {
    await this.pool.query(`
        UPDATE team_members SET accepted_at = NOW() WHERE team_id = $1 AND user_id = $2
    `, [teamId, userId]);
}

async allocateTokens(teamId: string, userId: string, tokens: number): Promise<void> {
    await this.pool.query(`
        UPDATE team_members SET token_allocation = $3 WHERE team_id = $1 AND user_id = $
    `, [teamId, userId, tokens]);
}

async recordUsage(teamId: string, userId: string, tokensUsed: number, model?: string): P
    const member = await this.getMember(teamId, userId);
```

```
        if (!member) throw new Error('Not a team member');

        // Check allocation if not shared pool
        const team = await this.getTeam(teamId);
        if (!team.shared_pool && member.token_allocation) {
            if (member.tokens_used_this_period + tokensUsed > member.token_allocation) {
                throw new Error('Token allocation exceeded');
            }
        }

        // Update member usage
        await this.pool.query(`
            UPDATE team_members SET tokens_used_this_period = tokens_used_this_period + $3
            WHERE team_id = $1 AND user_id = $2
        `, [teamId, userId, tokensUsed]);

        // Log usage
        await this.pool.query(`
            INSERT INTO team_usage_log (team_id, member_id, tokens_used, model)
            VALUES ($1, $2, $3, $4)
        `, [teamId, member.id, tokensUsed, model]);
    }

    async getTeamUsageStats(teamId: string): Promise<any> {
        const team = await this.getTeam(teamId);

        const members = await this.pool.query(`
            SELECT tm.*, u.email, u.display_name
            FROM team_members tm
            JOIN users u ON tm.user_id = u.id
            WHERE tm.team_id = $1 AND tm.is_active = true
        `, [teamId]);

        const totalUsed = members.rows.reduce((sum: number, m: any) => sum + (m.tokens_used_

        return {
            teamId,
            planName: team.plan_name,
            totalTokensMonthly: team.total_tokens_monthly,
            tokensUsed: totalUsed,
            tokensRemaining: team.total_tokens_monthly - totalUsed,
            members: members.rows.map((m: any) => ({
                userId: m.user_id,
                email: m.email,
                displayName: m.display_name,
                role: m.role,
```

```
                tokensUsed: m.tokens_used_this_period,
                allocation: m.token_allocation
            }))
        };
    }

    async resetPeriodUsage(teamId: string): Promise<void> {
        await this.pool.query(`
            UPDATE team_members SET tokens_used_this_period = 0 WHERE team_id = $1
        `, [teamId]);

        const now = new Date();
        const nextMonth = new Date(now.getFullYear(), now.getMonth() + 1, now.getDate());

        await this.pool.query(`
            UPDATE team_plans
            SET current_period_start = NOW(), current_period_end = $2
            WHERE id = $1
        `, [teamId, nextMonth]);
    }

    private async getTeam(teamId: string) {
        const result = await this.pool.query(`SELECT * FROM team_plans WHERE id = $1`, [tean
        return result.rows[0];
    }

    private async getMember(teamId: string, userId: string) {
        const result = await this.pool.query(
            `SELECT * FROM team_members WHERE team_id = $1 AND user_id = $2`,
            [teamId, userId]
        );
        return result.rows[0];
    }

    private async getMemberCount(teamId: string): Promise<number> {
        const result = await this.pool.query(
            `SELECT COUNT(*) FROM team_members WHERE team_id = $1 AND is_active = true`,
            [teamId]
        );
        return parseInt(result.rows[0].count);
    }
}
```

SECTION-28-ANALYTICS-INTEGRATION

SECTION 28: ANALYTICS INTEGRATION (v3.6.0)

28.1 Analytics Dashboard Extensions

```tsx
// apps/admin-dashboard/src/app/admin/analytics/page.tsx

'use client';

import React, { useState } from 'react';
import { useQuery } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from '@/components/
import { LineChart, Line, BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveCon
import { BarChart3, TrendingUp, Users, DollarSign, Cpu, Clock } from 'lucide-react';

export default function AnalyticsPage() {
    const [timeRange, setTimeRange] = useState('7d');
    const [metricType, setMetricType] = useState('usage');

    const { data: metrics } = useQuery({
        queryKey: ['analytics', timeRange, metricType],
        queryFn: async () => {
            const res = await fetch(`/api/admin/analytics?range=${timeRange}&type=${metricTy
            return res.json();
        }
    });

    const { data: modelStats } = useQuery({
        queryKey: ['model-stats', timeRange],
        queryFn: async () => {
            const res = await fetch(`/api/admin/analytics/models?range=${timeRange}`);
            return res.json();
        }
    });
```

```
const COLORS = ['#0088FE', '#00C49F', '#FFBB28', '#FF8042', '#8884D8'];

return (
    <div className="space-y-6">
        <div className="flex justify-between items-center">
            <h1 className="text-3xl font-bold">Analytics</h1>
            <div className="flex gap-4">
                <Select value={timeRange} onValueChange={setTimeRange}>
                    <SelectTrigger className="w-32">
                        <SelectValue />
                    </SelectTrigger>
                    <SelectContent>
                        <SelectItem value="24h">Last 24h</SelectItem>
                        <SelectItem value="7d">Last 7 days</SelectItem>
                        <SelectItem value="30d">Last 30 days</SelectItem>
                        <SelectItem value="90d">Last 90 days</SelectItem>
                    </SelectContent>
                </Select>
            </div>
        </div>

        {/* Summary Cards */}
        <div className="grid grid-cols-4 gap-4">
            <Card>
                <CardContent className="pt-6">
                    <div className="flex items-center justify-between">
                        <div>
                            <p className="text-sm text-gray-500">Total Requests</p>
                            <p className="text-2xl font-bold">{metrics?.totalRequests?.t
                            <p className="text-sm text-green-500">+{metrics?.requestsGro
                        </div>
                        <BarChart3 className="h-8 w-8 text-blue-500" />
                    </div>
                </CardContent>
            </Card>

            <Card>
                <CardContent className="pt-6">
                    <div className="flex items-center justify-between">
                        <div>
                            <p className="text-sm text-gray-500">Total Tokens</p>
                            <p className="text-2xl font-bold">{(metrics?.totalTokens / 1
                            <p className="text-sm text-green-500">+{metrics?.tokensGrowt
                        </div>
                        <Cpu className="h-8 w-8 text-purple-500" />
                    </div>
```

```
            </CardContent>
        </Card>

        <Card>
            <CardContent className="pt-6">
                <div className="flex items-center justify-between">
                    <div>
                        <p className="text-sm text-gray-500">Total Cost</p>
                        <p className="text-2xl font-bold">${metrics?.totalCost?.toFi
                        <p className="text-sm text-green-500">+{metrics?.costGrowth}
                    </div>
                    <DollarSign className="h-8 w-8 text-green-500" />
                </div>
            </CardContent>
        </Card>

        <Card>
            <CardContent className="pt-6">
                <div className="flex items-center justify-between">
                    <div>
                        <p className="text-sm text-gray-500">Avg Latency</p>
                        <p className="text-2xl font-bold">{metrics?.avgLatency}ms</p
                        <p className="text-sm text-green-500">-{metrics?.latencyImpr
                    </div>
                    <Clock className="h-8 w-8 text-orange-500" />
                </div>
            </CardContent>
        </Card>
    </div>

    {/* Charts */}
    <div className="grid grid-cols-2 gap-6">
        <Card>
            <CardHeader>
                <CardTitle>Usage Over Time</CardTitle>
            </CardHeader>
            <CardContent>
                <ResponsiveContainer width="100%" height={300}>
                    <LineChart data={metrics?.usageTimeSeries || []}>
                        <CartesianGrid strokeDasharray="3 3" />
                        <XAxis dataKey="date" />
                        <YAxis />
                        <Tooltip />
                        <Line type="monotone" dataKey="requests" stroke="#0088FE" />
                        <Line type="monotone" dataKey="tokens" stroke="#00C49F" />
                    </LineChart>
```

```
                </ResponsiveContainer>
            </CardContent>
        </Card>

        <Card>
            <CardHeader>
                <CardTitle>Model Distribution</CardTitle>
            </CardHeader>
            <CardContent>
                <ResponsiveContainer width="100%" height={300}>
                    <PieChart>
                        <Pie
                            data={modelStats?.distribution || []}
                            dataKey="value"
                            nameKey="name"
                            cx="50%"
                            cy="50%"
                            outerRadius={100}
                            label
                        >
                            {(modelStats?.distribution || []).map((entry: any, index
                                <Cell key={entry.name} fill={COLORS[index % COLORS.l
                            ))}
                        </Pie>
                        <Tooltip />
                    </PieChart>
                </ResponsiveContainer>
            </CardContent>
        </Card>
    </div>

    {/* Model Performance Table */}
    <Card>
        <CardHeader>
            <CardTitle>Model Performance</CardTitle>
        </CardHeader>
        <CardContent>
            <table className="w-full">
                <thead>
                    <tr className="border-b">
                        <th className="text-left py-2">Model</th>
                        <th className="text-right">Requests</th>
                        <th className="text-right">Avg Latency</th>
                        <th className="text-right">Success Rate</th>
                        <th className="text-right">Cost</th>
                    </tr>
```

```
                        </thead>
                        <tbody>
                            {(modelStats?.models || []).map((model: any) => (
                                <tr key={model.id} className="border-b">
                                    <td className="py-2">{model.name}</td>
                                    <td className="text-right">{model.requests.toLocaleStrin
                                    <td className="text-right">{model.avgLatency}ms</td>
                                    <td className="text-right">{(model.successRate * 100).to
                                    <td className="text-right">${model.cost.toFixed(2)}</td>
                                </tr>
                            ))}
                        </tbody>
                    </table>
                </CardContent>
            </Card>
        </div>
    );
}
```

SECTION-29-ADMIN-EXTENSIONS

SECTION 29: ADMIN DASHBOARD EXTENSIONS (v3.7.0)

NEW in v3.7.0: Extended admin dashboard capabilities for managing all v3.x features.

29.1 Admin Dashboard Navigation Update

```
// apps/admin-dashboard/src/components/layout/sidebar.tsx

const navigationItems = [
  // Existing items...
  { name: 'Dashboard', href: '/dashboard', icon: LayoutDashboard },
  { name: 'Tenants', href: '/tenants', icon: Building2 },
  { name: 'Users', href: '/users', icon: Users },
  { name: 'Administrators', href: '/administrators', icon: Shield },
```

```
  // AI & Models Section
  { type: 'separator', label: 'AI & Models' },
  { name: 'Providers', href: '/providers', icon: Cloud },
  { name: 'Models', href: '/models', icon: Brain },
  { name: 'Model Pricing', href: '/models/pricing', icon: DollarSign }, // NEW
  { name: 'Visual AI', href: '/visual-ai', icon: Image },
  { name: 'Thermal States', href: '/thermal-states', icon: Thermometer },

  // Think Tank Section (NEW)
  { type: 'separator', label: 'Think Tank' },
  { name: 'Think Tank Users', href: '/thinktank/users', icon: UserCircle },
  { name: 'Conversations', href: '/thinktank/conversations', icon: MessageSquare },
  { name: 'Domain Modes', href: '/thinktank/domain-modes', icon: Layers },
  { name: 'Model Categories', href: '/thinktank/model-categories', icon: Grid },

  // Operations Section
  { type: 'separator', label: 'Operations' },
  { name: 'Usage & Billing', href: '/billing', icon: Receipt },
  { name: 'Analytics', href: '/analytics', icon: BarChart },
  { name: 'Error Logs', href: '/errors', icon: AlertTriangle },
  { name: 'Audit Logs', href: '/audit', icon: FileText },

  // Settings Section
  { type: 'separator', label: 'Settings' },
  { name: 'Credentials', href: '/credentials', icon: Key },
  { name: 'System Config', href: '/config', icon: Settings },
];
```

29.2 Think Tank User Management Page

```
// apps/admin-dashboard/src/app/(dashboard)/thinktank/users/page.tsx

'use client';

import { useState } from 'react';
import { useQuery } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { DataTable } from '@/components/ui/data-table';
import { Badge } from '@/components/ui/badge';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Search, Download, UserX, Mail } from 'lucide-react';
import { formatDistanceToNow } from 'date-fns';
```

```
const columns = [
  {
    accessorKey: 'email',
    header: 'Email',
    cell: ({ row }) => (
      <div>
        <div className="font-medium">{row.original.email}</div>
        <div className="text-xs text-muted-foreground">
          ID: {row.original.id.slice(0, 8)}...
        </div>
      </div>
    ),
  },
  {
    accessorKey: 'display_name',
    header: 'Display Name',
  },
  {
    accessorKey: 'language',
    header: 'Language',
    cell: ({ row }) => (
      <Badge variant="outline">{row.original.language?.toUpperCase() || 'EN'}</Badge>
    ),
  },
  {
    accessorKey: 'subscription_tier',
    header: 'Tier',
    cell: ({ row }) => {
      const tier = row.original.subscription_tier;
      const colors = {
        free: 'bg-gray-100',
        pro: 'bg-blue-100 text-blue-800',
        team: 'bg-purple-100 text-purple-800',
        enterprise: 'bg-amber-100 text-amber-800',
      };
      return <Badge className={colors[tier] || colors.free}>{tier}</Badge>;
    },
  },
  {
    accessorKey: 'conversation_count',
    header: 'Conversations',
    cell: ({ row }) => row.original.conversation_count?.toLocaleString() || '0',
  },
  {
    accessorKey: 'total_tokens_used',
```

```
    header: 'Tokens Used',
    cell: ({ row }) => (row.original.total_tokens_used || 0).toLocaleString(),
  },
  {
    accessorKey: 'total_spent',
    header: 'Total Spent',
    cell: ({ row }) => `$${(row.original.total_spent || 0).toFixed(2)}`,
  },
  {
    accessorKey: 'last_active_at',
    header: 'Last Active',
    cell: ({ row }) =>
      row.original.last_active_at
        ? formatDistanceToNow(new Date(row.original.last_active_at), { addSuffix: true })
        : 'Never',
  },
  {
    accessorKey: 'status',
    header: 'Status',
    cell: ({ row }) => (
      <Badge variant={row.original.status === 'active' ? 'default' : 'destructive'}>
        {row.original.status}
      </Badge>
    ),
  },
];

export default function ThinkTankUsersPage() {
  const [search, setSearch] = useState('');

  const { data: users, isLoading } = useQuery({
    queryKey: ['thinktank-users', search],
    queryFn: () => fetch(`/api/admin/thinktank/users?search=${search}`).then(r => r.json()),
  });

  const { data: stats } = useQuery({
    queryKey: ['thinktank-user-stats'],
    queryFn: () => fetch('/api/admin/thinktank/users/stats').then(r => r.json()),
  });

  return (
    <div className="space-y-6">
      <div className="flex justify-between items-center">
        <div>
          <h1 className="text-2xl font-bold">Think Tank Users</h1>
          <p className="text-muted-foreground">
```

```
      Manage Think Tank consumer users
    </p>
  </div>
  <Button variant="outline">
    <Download className="h-4 w-4 mr-2" />
    Export
  </Button>
</div>

{/* Stats Cards */}
<div className="grid gap-4 md:grid-cols-4">
  <Card>
    <CardHeader className="pb-2">
      <CardTitle className="text-sm font-medium text-muted-foreground">
        Total Users
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="text-2xl font-bold">{stats?.totalUsers?.toLocaleString() || 0}</
    </CardContent>
  </Card>
  <Card>
    <CardHeader className="pb-2">
      <CardTitle className="text-sm font-medium text-muted-foreground">
        Active (7d)
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="text-2xl font-bold">{stats?.activeUsers7d?.toLocaleString() || 
    </CardContent>
  </Card>
  <Card>
    <CardHeader className="pb-2">
      <CardTitle className="text-sm font-medium text-muted-foreground">
        Pro+ Subscribers
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="text-2xl font-bold">{stats?.paidUsers?.toLocaleString() || 0}</d
    </CardContent>
  </Card>
  <Card>
    <CardHeader className="pb-2">
      <CardTitle className="text-sm font-medium text-muted-foreground">
        Total Revenue
      </CardTitle>
```

```
            </CardHeader>
            <CardContent>
              <div className="text-2xl font-bold">${stats?.totalRevenue?.toLocaleString() || (
            </CardContent>
          </Card>
        </div>

        {/* Search & Table */}
        <Card>
          <CardHeader>
            <div className="flex items-center gap-4">
              <div className="relative flex-1 max-w-sm">
                <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-muted
                <Input
                  placeholder="Search by email or name..."
                  value={search}
                  onChange={(e) => setSearch(e.target.value)}
                  className="pl-9"
                />
              </div>
            </div>
          </CardHeader>
          <CardContent>
            <DataTable
              columns={columns}
              data={users?.data || []}
              isLoading={isLoading}
            />
          </CardContent>
        </Card>
      </div>
    );
}
```

29.3 Domain Modes Configuration Page

```
// apps/admin-dashboard/src/app/(dashboard)/thinktank/domain-modes/page.tsx

'use client';

import { useState } from 'react';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle, CardDescription } from '@/components/ui/c
import { Switch } from '@/components/ui/switch';
```

```tsx
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Label } from '@/components/ui/label';
import { Textarea } from '@/components/ui/textarea';
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from '@/components/
import { Badge } from '@/components/ui/badge';
import { toast } from 'sonner';
import {
  Stethoscope, Scale, Code2, Lightbulb, GraduationCap,
  PenTool, FlaskConical, Save
} from 'lucide-react';

const DOMAIN_MODES = [
  { id: 'general', name: 'General', icon: Lightbulb, description: 'Default mode for general
  { id: 'medical', name: 'Medical', icon: Stethoscope, description: 'Healthcare and medical
  { id: 'legal', name: 'Legal', icon: Scale, description: 'Legal research and analysis' },
  { id: 'code', name: 'Code', icon: Code2, description: 'Programming and development' },
  { id: 'academic', name: 'Academic', icon: GraduationCap, description: 'Research and educat
  { id: 'creative', name: 'Creative', icon: PenTool, description: 'Writing and content creat
  { id: 'scientific', name: 'Scientific', icon: FlaskConical, description: 'Scientific resea
];

export default function DomainModesPage() {
  const queryClient = useQueryClient();

  const { data: config } = useQuery({
    queryKey: ['domain-modes-config'],
    queryFn: () => fetch('/api/admin/thinktank/domain-modes').then(r => r.json()),
  });

  const { data: models } = useQuery({
    queryKey: ['available-models'],
    queryFn: () => fetch('/api/admin/models?enabled=true').then(r => r.json()),
  });

  const updateMutation = useMutation({
    mutationFn: (data: any) =>
      fetch('/api/admin/thinktank/domain-modes', {
        method: 'PUT',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data),
      }).then(r => r.json()),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['domain-modes-config'] });
      toast.success('Domain mode configuration saved');
    },
```

```
});

const [localConfig, setLocalConfig] = useState(config || {});

return (
  <div className="space-y-6">
    <div className="flex justify-between items-center">
      <div>
        <h1 className="text-2xl font-bold">Domain Modes</h1>
        <p className="text-muted-foreground">
          Configure specialized AI modes for different use cases
        </p>
      </div>
      <Button onClick={() => updateMutation.mutate(localConfig)}>
        <Save className="h-4 w-4 mr-2" />
        Save Changes
      </Button>
    </div>

    <div className="grid gap-4">
      {DOMAIN_MODES.map((mode) => {
        const ModeIcon = mode.icon;
        const modeConfig = localConfig?.modes?.[mode.id] || {};

        return (
          <Card key={mode.id}>
            <CardHeader>
              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <div className="p-2 rounded-lg bg-primary/10">
                    <ModeIcon className="h-5 w-5 text-primary" />
                  </div>
                  <div>
                    <CardTitle className="text-lg">{mode.name}</CardTitle>
                    <CardDescription>{mode.description}</CardDescription>
                  </div>
                </div>
                <Switch
                  checked={modeConfig.enabled !== false}
                  onCheckedChange={(checked) =>
                    setLocalConfig({
                      ...localConfig,
                      modes: {
                        ...localConfig.modes,
                        [mode.id]: { ...modeConfig, enabled: checked },
                      },
```

```
        })
      }
    />
  </div>
</CardHeader>
<CardContent className="space-y-4">
  <div className="grid gap-4 md:grid-cols-2">
    <div className="space-y-2">
      <Label>Default Model</Label>
      <Select
        value={modeConfig.defaultModel || 'auto'}
        onValueChange={(value) =>
          setLocalConfig({
            ...localConfig,
            modes: {
              ...localConfig.modes,
              [mode.id]: { ...modeConfig, defaultModel: value },
            },
          })
        }
      >
        <SelectTrigger>
          <SelectValue placeholder="Auto (RADIANT Brain)" />
        </SelectTrigger>
        <SelectContent>
          <SelectItem value="auto">Auto (RADIANT Brain)</SelectItem>
          {(models?.data || []).map((model: any) => (
            <SelectItem key={model.id} value={model.id}>
              {model.display_name}
            </SelectItem>
          ))}
        </SelectContent>
      </Select>
    </div>

    <div className="space-y-2">
      <Label>Temperature</Label>
      <Input
        type="number"
        min="0"
        max="2"
        step="0.1"
        value={modeConfig.temperature || 0.7}
        onChange={(e) =>
          setLocalConfig({
            ...localConfig,
```

```
                        modes: {
                          ...localConfig.modes,
                          [mode.id]: { ...modeConfig, temperature: parseFloat(e.target.val
                        },
                      })
                    }
                  />
                </div>
              </div>

              <div className="space-y-2">
                <Label>System Prompt Override</Label>
                <Textarea
                  placeholder="Optional: Custom system prompt for this mode..."
                  value={modeConfig.systemPrompt || ''}
                  onChange={(e) =>
                    setLocalConfig({
                      ...localConfig,
                      modes: {
                        ...localConfig.modes,
                        [mode.id]: { ...modeConfig, systemPrompt: e.target.value },
                      },
                    })
                  }
                  rows={3}
                />
              </div>
            </CardContent>
          </Card>
        );
      })}
    </div>
  </div>
);
}
```

SECTION-30-DYNAMIC-PROVIDER-REGISTRY

SECTION 30: DYNAMIC PROVIDER REGISTRY + xAI/GROK (v3.7.0)

NEW in v3.7.0: Database-driven model registry with automatic sync and xAI/Grok integration.

30.1 Complete Provider & Model Registry

External AI Providers (21+)

Provider ID	Display Name	API Base	Auth Type
`anthropic`	Anthropic	api.anthropic.com	API Key
`openai`	OpenAI	api.openai.com	API Key
`google`	Google (Gemini)	generativelanguage.googleapis.com	API Key
`xai`	xAI (Grok)	api.x.ai	API Key
`mistral`	Mistral AI	api.mistral.ai	API Key
`cohere`	Cohere	api.cohere.ai	API Key
`perplexity`	Perplexity	api.perplexity.ai	API Key
`deepseek`	DeepSeek	api.deepseek.com	API Key
`together`	Together AI	api.together.xyz	API Key
`fireworks`	Fireworks AI	api.fireworks.ai	API Key
`groq`	Groq	api.groq.com	API Key
`replicate`	Replicate	api.replicate.com	API Key
`huggingface`	Hugging Face	api-inference.huggingface.co	API Key
`bedrock`	AWS Bedrock	bedrock-runtime.amazonaws.com	IAM
`azure_openai`	Azure OpenAI	*.openai.azure.com	API Key
`vertex_ai`	Google Vertex AI	*.aiplatform.googleapis.com	Service Account

---

30.2 xAI/Grok Models (10 Models)

Model ID	Display Name	Context	Input $/1M	Output $/1M	Capabilities
`grok-3`	Grok 3	131K	$3.00	$15.00	Flagship, real-time info
`grok-3-fast`	Grok 3 Fast	131K	$1.00	$5.00	Speed-optimized
`grok-3-mini`	Grok 3 Mini	131K	$0.30	$1.50	Cost-effective
`grok-2`	Grok 2	131K	$2.00	$10.00	Previous generation
`grok-2-vision`	Grok 2 Vision	32K	$2.00	$10.00	Image understanding
`grok-2-mini`	Grok 2 Mini	131K	$0.20	$1.00	Budget option
`grok-coder`	Grok Coder	131K	$1.50	$7.50	Code generation
`grok-analyst`	Grok Analyst	131K	$2.00	$10.00	Data analysis
`grok-embed`	Grok Embed	8K	$0.10	-	Text embeddings
`grok-realtime`	Grok Realtime	32K	$5.00	$20.00	Voice/streaming

xAI Provider Configuration

```typescript
// packages/shared/src/providers/xai.ts

export const XAI_PROVIDER_CONFIG = {
  id: 'xai',
  displayName: 'xAI (Grok)',
  apiBase: 'https://api.x.ai/v1',
  authType: 'api_key',
  authHeader: 'Authorization',
  authPrefix: 'Bearer ',

  models: [
    {
      id: 'grok-3',
      displayName: 'Grok 3',
      contextWindow: 131072,
      maxOutputTokens: 8192,
      supportedModalities: ['text'],
      pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
      capabilities: ['chat', 'reasoning', 'analysis', 'realtime_info'],
```

```
  isNovel: false,
},
{
  id: 'grok-3-fast',
  displayName: 'Grok 3 Fast',
  contextWindow: 131072,
  maxOutputTokens: 8192,
  supportedModalities: ['text'],
  pricing: { inputPer1M: 1.00, outputPer1M: 5.00 },
  capabilities: ['chat', 'fast_response'],
  isNovel: false,
},
{
  id: 'grok-3-mini',
  displayName: 'Grok 3 Mini',
  contextWindow: 131072,
  maxOutputTokens: 4096,
  supportedModalities: ['text'],
  pricing: { inputPer1M: 0.30, outputPer1M: 1.50 },
  capabilities: ['chat', 'cost_effective'],
  isNovel: false,
},
{
  id: 'grok-2',
  displayName: 'Grok 2',
  contextWindow: 131072,
  maxOutputTokens: 8192,
  supportedModalities: ['text'],
  pricing: { inputPer1M: 2.00, outputPer1M: 10.00 },
  capabilities: ['chat', 'reasoning'],
  isNovel: false,
},
{
  id: 'grok-2-vision',
  displayName: 'Grok 2 Vision',
  contextWindow: 32768,
  maxOutputTokens: 4096,
  supportedModalities: ['text', 'image'],
  pricing: { inputPer1M: 2.00, outputPer1M: 10.00 },
  capabilities: ['chat', 'vision', 'image_analysis'],
  isNovel: true,
},
{
  id: 'grok-2-mini',
  displayName: 'Grok 2 Mini',
  contextWindow: 131072,
```

```
  maxOutputTokens: 4096,
  supportedModalities: ['text'],
  pricing: { inputPer1M: 0.20, outputPer1M: 1.00 },
  capabilities: ['chat'],
  isNovel: false,
},
{
  id: 'grok-coder',
  displayName: 'Grok Coder',
  contextWindow: 131072,
  maxOutputTokens: 8192,
  supportedModalities: ['text'],
  pricing: { inputPer1M: 1.50, outputPer1M: 7.50 },
  capabilities: ['chat', 'code_generation', 'code_review'],
  isNovel: true,
},
{
  id: 'grok-analyst',
  displayName: 'Grok Analyst',
  contextWindow: 131072,
  maxOutputTokens: 8192,
  supportedModalities: ['text'],
  pricing: { inputPer1M: 2.00, outputPer1M: 10.00 },
  capabilities: ['chat', 'data_analysis', 'insights'],
  isNovel: true,
},
{
  id: 'grok-embed',
  displayName: 'Grok Embed',
  contextWindow: 8192,
  maxOutputTokens: 0,
  supportedModalities: ['text'],
  pricing: { inputPer1M: 0.10, outputPer1M: 0 },
  capabilities: ['embeddings'],
  isNovel: false,
},
{
  id: 'grok-realtime',
  displayName: 'Grok Realtime',
  contextWindow: 32768,
  maxOutputTokens: 4096,
  supportedModalities: ['text', 'audio'],
  pricing: { inputPer1M: 5.00, outputPer1M: 20.00 },
  capabilities: ['chat', 'voice', 'streaming', 'realtime'],
  isNovel: true,
},
```

```
  ],
};
```

30.3 Complete Model Registry

All External Models (60+)

```
// packages/shared/src/models/registry.ts

export const MODEL_REGISTRY: ModelDefinition[] = [
  //
  // ANTHROPIC MODELS
  //
  {
    id: 'claude-4-opus',
    providerId: 'anthropic',
    displayName: 'Claude 4 Opus',
    contextWindow: 200000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 15.00, outputPer1M: 75.00 },
    capabilities: ['chat', 'reasoning', 'analysis', 'code', 'vision'],
    isNovel: false,
    category: 'flagship',
  },
  {
    id: 'claude-4-sonnet',
    providerId: 'anthropic',
    displayName: 'Claude 4 Sonnet',
    contextWindow: 200000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
    capabilities: ['chat', 'reasoning', 'analysis', 'code', 'vision'],
    isNovel: false,
    category: 'balanced',
  },
  {
    id: 'claude-3.5-haiku',
    providerId: 'anthropic',
    displayName: 'Claude 3.5 Haiku',
    contextWindow: 200000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.25, outputPer1M: 1.25 },
    capabilities: ['chat', 'code'],
    isNovel: false,
    category: 'economy',
```

```
},

//
// OPENAI MODELS
//
{
  id: 'gpt-4o',
  providerId: 'openai',
  displayName: 'GPT-4o',
  contextWindow: 128000,
  maxOutputTokens: 16384,
  pricing: { inputPer1M: 2.50, outputPer1M: 10.00 },
  capabilities: ['chat', 'reasoning', 'vision', 'audio'],
  isNovel: false,
  category: 'flagship',
},
{
  id: 'gpt-4o-mini',
  providerId: 'openai',
  displayName: 'GPT-4o Mini',
  contextWindow: 128000,
  maxOutputTokens: 16384,
  pricing: { inputPer1M: 0.15, outputPer1M: 0.60 },
  capabilities: ['chat', 'vision'],
  isNovel: false,
  category: 'economy',
},
{
  id: 'o1',
  providerId: 'openai',
  displayName: 'o1 Reasoning',
  contextWindow: 200000,
  maxOutputTokens: 100000,
  pricing: { inputPer1M: 15.00, outputPer1M: 60.00 },
  capabilities: ['reasoning', 'analysis', 'math', 'code'],
  isNovel: false,
  category: 'specialized',
},
{
  id: 'o1-pro',
  providerId: 'openai',
  displayName: 'o1 Pro',
  contextWindow: 200000,
  maxOutputTokens: 100000,
  pricing: { inputPer1M: 150.00, outputPer1M: 600.00 },
  capabilities: ['reasoning', 'analysis', 'math', 'code', 'extended_thinking'],
```

```
  isNovel: true,
  category: 'novel',
},
{
  id: 'gpt-4o-realtime',
  providerId: 'openai',
  displayName: 'GPT-4o Realtime',
  contextWindow: 128000,
  maxOutputTokens: 4096,
  pricing: { inputPer1M: 5.00, outputPer1M: 20.00 },
  capabilities: ['voice', 'streaming', 'realtime'],
  isNovel: true,
  category: 'novel',
},

//
// GOOGLE MODELS
//
{
  id: 'gemini-2.0-pro',
  providerId: 'google',
  displayName: 'Gemini 2.0 Pro',
  contextWindow: 2000000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 1.25, outputPer1M: 5.00 },
  capabilities: ['chat', 'reasoning', 'vision', 'code'],
  isNovel: false,
  category: 'flagship',
},
{
  id: 'gemini-2.0-flash',
  providerId: 'google',
  displayName: 'Gemini 2.0 Flash',
  contextWindow: 1000000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 0.075, outputPer1M: 0.30 },
  capabilities: ['chat', 'vision', 'fast'],
  isNovel: false,
  category: 'balanced',
},
{
  id: 'gemini-2.0-ultra',
  providerId: 'google',
  displayName: 'Gemini 2.0 Ultra',
  contextWindow: 2000000,
  maxOutputTokens: 8192,
```

```
  pricing: { inputPer1M: 5.00, outputPer1M: 15.00 },
  capabilities: ['chat', 'reasoning', 'vision', 'multimodal'],
  isNovel: true,
  category: 'novel',
},
{
  id: 'gemini-2.0-pro-exp',
  providerId: 'google',
  displayName: 'Gemini Pro Experimental',
  contextWindow: 10000000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 2.50, outputPer1M: 10.00 },
  capabilities: ['chat', 'reasoning', 'massive_context'],
  isNovel: true,
  category: 'novel',
},

//
// XAI/GROK MODELS (from 30.2)
//
{
  id: 'grok-3',
  providerId: 'xai',
  displayName: 'Grok 3',
  contextWindow: 131072,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
  capabilities: ['chat', 'reasoning', 'realtime_info'],
  isNovel: false,
  category: 'flagship',
},
{
  id: 'grok-3-fast',
  providerId: 'xai',
  displayName: 'Grok 3 Fast',
  contextWindow: 131072,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 1.00, outputPer1M: 5.00 },
  capabilities: ['chat', 'fast'],
  isNovel: false,
  category: 'balanced',
},
{
  id: 'grok-3-mini',
  providerId: 'xai',
  displayName: 'Grok 3 Mini',
```

```
  contextWindow: 131072,
  maxOutputTokens: 4096,
  pricing: { inputPer1M: 0.30, outputPer1M: 1.50 },
  capabilities: ['chat'],
  isNovel: false,
  category: 'economy',
},

//
// DEEPSEEK MODELS
//
{
  id: 'deepseek-v3',
  providerId: 'deepseek',
  displayName: 'DeepSeek V3',
  contextWindow: 64000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 0.14, outputPer1M: 0.28 },
  capabilities: ['chat', 'code', 'reasoning'],
  isNovel: false,
  category: 'economy',
},
{
  id: 'deepseek-r1',
  providerId: 'deepseek',
  displayName: 'DeepSeek R1',
  contextWindow: 64000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 0.55, outputPer1M: 2.19 },
  capabilities: ['reasoning', 'chain_of_thought', 'analysis'],
  isNovel: true,
  category: 'novel',
},

//
// MISTRAL MODELS
//
{
  id: 'mistral-large-2',
  providerId: 'mistral',
  displayName: 'Mistral Large 2',
  contextWindow: 128000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 2.00, outputPer1M: 6.00 },
  capabilities: ['chat', 'reasoning', 'multilingual'],
  isNovel: false,
```

```
  category: 'specialized',
},
{
  id: 'codestral-latest',
  providerId: 'mistral',
  displayName: 'Codestral',
  contextWindow: 32000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 0.30, outputPer1M: 0.90 },
  capabilities: ['code', 'code_generation', 'code_review'],
  isNovel: false,
  category: 'specialized',
},

//
// PERPLEXITY MODELS
//
{
  id: 'perplexity-sonar-pro',
  providerId: 'perplexity',
  displayName: 'Sonar Pro',
  contextWindow: 128000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
  capabilities: ['search', 'chat', 'citations', 'realtime_info'],
  isNovel: false,
  category: 'specialized',
},

//
// NOVEL/EXPERIMENTAL MODELS
//
{
  id: 'claude-4-opus-agents',
  providerId: 'anthropic',
  displayName: 'Claude Opus Agents',
  contextWindow: 200000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 15.00, outputPer1M: 75.00 },
  capabilities: ['chat', 'tool_use', 'computer_use', 'agents'],
  isNovel: true,
  category: 'novel',
},
{
  id: 'qwen-2.5-coder',
  providerId: 'together',
```

```
    displayName: 'Qwen 2.5 Coder',
    contextWindow: 128000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.30, outputPer1M: 0.90 },
    capabilities: ['code', 'code_generation'],
    isNovel: true,
    category: 'novel',
  },
  {
    id: 'llama-3.3-70b',
    providerId: 'together',
    displayName: 'Llama 3.3 70B',
    contextWindow: 128000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.88, outputPer1M: 0.88 },
    capabilities: ['chat', 'reasoning', 'open_weights'],
    isNovel: true,
    category: 'novel',
  },
  {
    id: 'phi-4',
    providerId: 'azure_openai',
    displayName: 'Phi-4',
    contextWindow: 16000,
    maxOutputTokens: 4096,
    pricing: { inputPer1M: 0.07, outputPer1M: 0.14 },
    capabilities: ['chat', 'reasoning', 'efficient'],
    isNovel: true,
    category: 'novel',
  },
];
```

30.4 Provider Sync Lambda

```
// packages/lambdas/src/handlers/admin/sync-providers.ts

import { ScheduledHandler } from 'aws-lambda';
import { Pool } from 'pg';
import { MODEL_REGISTRY } from '@radiant/shared';
import { createLogger } from '@radiant/shared';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });
const logger = createLogger('provider-sync');
```

```
export const handler: ScheduledHandler = async () => {
  logger.info('Starting provider/model sync');

  const client = await pool.connect();

  try {
    await client.query('BEGIN');

    // Sync all models from registry
    for (const model of MODEL_REGISTRY) {
      await client.query(`
        INSERT INTO models (
          id, provider_id, display_name, model_type, context_window,
          max_output_tokens, pricing, capabilities, is_novel, category,
          is_enabled, updated_at
        ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, TRUE, NOW())
        ON CONFLICT (id) DO UPDATE SET
          display_name = EXCLUDED.display_name,
          context_window = EXCLUDED.context_window,
          max_output_tokens = EXCLUDED.max_output_tokens,
          capabilities = EXCLUDED.capabilities,
          is_novel = EXCLUDED.is_novel,
          category = EXCLUDED.category,
          updated_at = NOW()
        -- Note: pricing is NOT updated to preserve admin overrides
      `, [
        model.id,
        model.providerId,
        model.displayName,
        'chat',
        model.contextWindow,
        model.maxOutputTokens,
        JSON.stringify(model.pricing),
        JSON.stringify(model.capabilities),
        model.isNovel || false,
        model.category || 'general',
      ]);
    }

    await client.query('COMMIT');
    logger.info(`Synced ${MODEL_REGISTRY.length} models`);

  } catch (error) {
    await client.query('ROLLBACK');
    logger.error('Sync failed', { error });
    throw error;
```

```
  } finally {
    client.release();
  }
};
```

SECTION-31-MODEL-SELECTION-PRICING

SECTION 31: THINK TANK MODEL SELECTION & EDITABLE PRICING (v3.8.0)

NEW in v3.8.0: Users can now manually select AI models in Think Tank. All pricing is admin-editable with bulk controls and individual overrides.

31.1 Model Categories for Think Tank

Standard Models (15) - Production-Ready

ID	Display Name	Provider	Context	Input $/1M	Output $/1M	Best For
`claude-4-opus`	Claude 4 Opus	Anthropic	200K	$15.00	$75.00	Complex reasoning, analysis
`claude-4-sonnet`	Claude 4 Sonnet	Anthropic	200K	$3.00	$15.00	Quality/cost balance
`claude-3-5-haiku`	Claude 3.5 Haiku	Anthropic	200K	$0.25	$1.25	Fast responses
`gpt-4o`	GPT-4o	OpenAI	128K	$2.50	$10.00	Multimodal, reliable
`gpt-4o-mini`	GPT-4o Mini	OpenAI	128K	$0.15	$0.60	Cost-effective
`o1`	o1 Reasoning	OpenAI	200K	$15.00	$60.00	Multi-step reasoning

ID	Display Name	Provider	Context	Input $/1M	Output $/1M	Best For
gemini-2.0-pro	Gemini 2.0 Pro	Google	2M	$1.25	$5.00	Massive context
gemini-2.0-flash	Gemini 2.0 Flash	Google	1M	$0.075	$0.30	Speed, large context
grok-3	Grok 3	xAI	131K	$3.00	$15.00	Real-time info
grok-3-fast	Grok 3 Fast	xAI	131K	$1.00	$5.00	Quick responses
grok-3-mini	Grok 3 Mini	xAI	131K	$0.30	$1.50	Budget Grok
deepseek-v3	DeepSeek V3	DeepSeek	64K	$0.14	$0.28	Extremely low cost
mistral-large-2	Mistral Large 2	Mistral	128K	$2.00	$6.00	Multilingual
codestral-latest	Codestral	Mistral	32K	$0.30	$0.90	Code generation
perplexity-sonar-pro	Sonar Pro	Perplexity	128K	$3.00	$15.00	Web search

Novel Models (15) - Cutting-Edge/Experimental

ID	Display Name	Provider	Context	Input $/1M	Output $/1M	Novel Feature
o1-pro	o1 Pro	OpenAI	200K	$150.00	$600.00	Extended reasoning chains
deepseek-r1	DeepSeek R1	DeepSeek	64K	$0.55	$2.19	Open reasoning model
gemini-2.0-ultra	Gemini 2.0 Ultra	Google	2M	$5.00	$15.00	Native multimodal
gemini-2.0-pro-exp	Gemini 2.0 Pro Exp	Google	10M	$2.50	$10.00	10M token context
grok-2-vision	Grok 2 Vision	xAI	32K	$2.00	$10.00	Image understanding
grok-realtime	Grok Realtime	xAI	32K	$5.00	$20.00	Live streaming

ID	Display Name	Provider	Context	Input $/1M	Output $/1M	Novel Feature
grok-code	Grok Coder	xAI	131K	$1.50	$7.50	Specialized code gen
grok-analyst	Grok Analyst	xAI	131K	$2.00	$10.00	Data analysis
gpt-4o-realtime	GPT-4o Realtime	OpenAI	128K	$5.00	$20.00	Voice/video streaming
claude-opus-agents	Claude Opus Agents	Anthropic	200K	$15.00	$75.00	Tool use, computer use
qwen-2.5-coder	Qwen 2.5 Coder	Together	128K	$0.30	$0.90	Open-source code
llama-3.3-70b	Llama 3.3 70B	Together	128K	$0.88	$0.88	Open weights
phi-4	Phi-4	Microsoft	16K	$0.07	$0.14	Small but capable
grok-embed	Grok Embed	xAI	8K	$0.10	-	Text embeddings
command-r-plus	Command R+	Cohere	128K	$2.50	$10.00	RAG optimized

---

31.2 Database Schema for Model Selection

```sql
-- Migration: 20241223_031_thinktank_model_selection.sql

-- Add model categorization columns
ALTER TABLE models ADD COLUMN IF NOT EXISTS is_novel BOOLEAN DEFAULT FALSE;
ALTER TABLE models ADD COLUMN IF NOT EXISTS category VARCHAR(50) DEFAULT 'general';
ALTER TABLE models ADD COLUMN IF NOT EXISTS thinktank_enabled BOOLEAN DEFAULT TRUE;
ALTER TABLE models ADD COLUMN IF NOT EXISTS thinktank_display_order INTEGER DEFAULT 100;

-- User model preferences for Think Tank
CREATE TABLE IF NOT EXISTS thinktank_user_model_preferences (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES thinktank_users(id) ON DELETE CASCADE,
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Selection Mode: 'auto' | 'manual' | 'favorites'
    selection_mode VARCHAR(20) NOT NULL DEFAULT 'auto',

    -- Default Model (when manual mode)
    default_model_id VARCHAR(100),
```

```
    -- Favorite Models (JSON array of model IDs)
    favorite_models JSONB DEFAULT '[]'::JSONB,

    -- Category Preferences
    show_standard_models BOOLEAN DEFAULT TRUE,
    show_novel_models BOOLEAN DEFAULT TRUE,
    show_self_hosted_models BOOLEAN DEFAULT FALSE,

    -- Cost Preferences
    show_cost_per_message BOOLEAN DEFAULT TRUE,
    max_cost_per_message DECIMAL(10, 6), -- NULL = no limit
    prefer_cost_optimization BOOLEAN DEFAULT FALSE,

    -- Domain Mode Model Overrides
    -- Example: {"medical": "claude-4-opus", "code": "codestral-latest"}
    domain_mode_model_overrides JSONB DEFAULT '{}'::JSONB,

    -- Recent Models (for quick access)
    recent_models JSONB DEFAULT '[]'::JSONB,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(user_id)
);

CREATE INDEX idx_thinktank_model_prefs_user ON thinktank_user_model_preferences(user_id);
CREATE INDEX idx_thinktank_model_prefs_tenant ON thinktank_user_model_preferences(tenant_id)

-- Trigger for updated_at
CREATE TRIGGER update_thinktank_model_prefs_timestamp
    BEFORE UPDATE ON thinktank_user_model_preferences
    FOR EACH ROW
    EXECUTE FUNCTION update_updated_at_column();
```

31.3 Admin Editable Pricing Schema

```
-- Migration: 20241223_032_editable_pricing.sql

-- Pricing configuration table (admin-editable)
CREATE TABLE IF NOT EXISTS pricing_config (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
```

```
    -- Global Markup Defaults
    external_default_markup DECIMAL(5, 4) DEFAULT 0.40,      -- 40%
    self_hosted_default_markup DECIMAL(5, 4) DEFAULT 0.75, -- 75%

    -- Minimum Charges
    minimum_charge_per_request DECIMAL(10, 6) DEFAULT 0.001,

    -- Grace Period for Price Increases (hours)
    price_increase_grace_period_hours INTEGER DEFAULT 24,

    -- Auto-Update Settings
    auto_update_from_providers BOOLEAN DEFAULT TRUE,
    auto_update_frequency VARCHAR(20) DEFAULT 'daily', -- 'hourly', 'daily', 'weekly'
    last_auto_update TIMESTAMPTZ,

    -- Notification Settings
    notify_on_price_change BOOLEAN DEFAULT TRUE,
    notify_threshold_percent DECIMAL(5, 2) DEFAULT 10.00, -- Notify if price changes >10%

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tenant_id)
);

-- Model-specific pricing overrides (admin-editable per model)
CREATE TABLE IF NOT EXISTS model_pricing_overrides (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    model_id VARCHAR(100) NOT NULL,

    -- Override Values (NULL = use defaults)
    markup_override DECIMAL(5, 4),           -- Override markup percentage
    input_price_override DECIMAL(12, 6),     -- Override input price per 1M tokens
    output_price_override DECIMAL(12, 6),    -- Override output price per 1M tokens

    -- Effective Dates (for scheduled price changes)
    effective_from TIMESTAMPTZ DEFAULT NOW(),
    effective_to TIMESTAMPTZ,

    -- Audit
    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
```

```
    UNIQUE(tenant_id, model_id, effective_from)
);

CREATE INDEX idx_pricing_overrides_tenant ON model_pricing_overrides(tenant_id);
CREATE INDEX idx_pricing_overrides_model ON model_pricing_overrides(model_id);
CREATE INDEX idx_pricing_overrides_effective ON model_pricing_overrides(effective_from, effe

-- Price history for auditing
CREATE TABLE IF NOT EXISTS price_history (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    model_id VARCHAR(100) NOT NULL,

    -- Previous Values
    previous_input_price DECIMAL(12, 6),
    previous_output_price DECIMAL(12, 6),
    previous_markup DECIMAL(5, 4),

    -- New Values
    new_input_price DECIMAL(12, 6),
    new_output_price DECIMAL(12, 6),
    new_markup DECIMAL(5, 4),

    -- Change Source
    change_source VARCHAR(50), -- 'admin', 'auto_sync', 'bulk_update'
    changed_by UUID REFERENCES administrators(id),

    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_price_history_tenant_model ON price_history(tenant_id, model_id);
CREATE INDEX idx_price_history_date ON price_history(created_at);

-- View for effective pricing (combines base + overrides)
CREATE OR REPLACE VIEW effective_model_pricing AS
SELECT
    m.id AS model_id,
    m.display_name,
    m.provider_id,
    m.is_novel,
    m.category,
    m.thinktank_enabled,

    -- Base Prices
    COALESCE(m.pricing->>'input_tokens', '0')::DECIMAL AS base_input_price,
    COALESCE(m.pricing->>'output_tokens', '0')::DECIMAL AS base_output_price,
```

```sql
    -- Effective Markup (override > category default > global default)
    COALESCE(
        mpo.markup_override,
        CASE
            WHEN m.provider_id = 'self_hosted' THEN pc.self_hosted_default_markup
            ELSE pc.external_default_markup
        END,
        0.40
    ) AS effective_markup,

    -- Final User Prices (with markup)
    ROUND(
        COALESCE(mpo.input_price_override, COALESCE(m.pricing->>'input_tokens', '0')::DECIMA
        (1 + COALESCE(mpo.markup_override,
            CASE WHEN m.provider_id = 'self_hosted' THEN pc.self_hosted_default_markup ELSE
            0.40)),
        6
    ) AS user_input_price,

    ROUND(
        COALESCE(mpo.output_price_override, COALESCE(m.pricing->>'output_tokens', '0')::DECI
        (1 + COALESCE(mpo.markup_override,
            CASE WHEN m.provider_id = 'self_hosted' THEN pc.self_hosted_default_markup ELSE
            0.40)),
        6
    ) AS user_output_price,

    -- Override Status
    mpo.id IS NOT NULL AS has_override,
    mpo.effective_from,
    mpo.effective_to

FROM models m
LEFT JOIN pricing_config pc ON TRUE
LEFT JOIN model_pricing_overrides mpo ON m.id = mpo.model_id
    AND (mpo.effective_from <= NOW() OR mpo.effective_from IS NULL)
    AND (mpo.effective_to > NOW() OR mpo.effective_to IS NULL);
```

31.4 Admin Pricing Dashboard

```tsx
// apps/admin-dashboard/src/app/(dashboard)/models/pricing/page.tsx

'use client';
```

```tsx
import { useState } from 'react';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle, CardDescription } from '@/components/ui/c
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Label } from '@/components/ui/label';
import { Switch } from '@/components/ui/switch';
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';
import { Badge } from '@/components/ui/badge';
import { Slider } from '@/components/ui/slider';
import {
  Table, TableBody, TableCell, TableHead, TableHeader, TableRow
} from '@/components/ui/table';
import {
  Dialog, DialogContent, DialogHeader, DialogTitle, DialogTrigger, DialogFooter
} from '@/components/ui/dialog';
import { toast } from 'sonner';
import { Save, RefreshCw, DollarSign, Percent, History, Edit2 } from 'lucide-react';

interface PricingConfig {
  externalDefaultMarkup: number;
  selfHostedDefaultMarkup: number;
  minimumChargePerRequest: number;
  priceIncreaseGracePeriodHours: number;
  autoUpdateFromProviders: boolean;
  autoUpdateFrequency: 'hourly' | 'daily' | 'weekly';
  notifyOnPriceChange: boolean;
  notifyThresholdPercent: number;
}

interface ModelPricing {
  modelId: string;
  displayName: string;
  providerId: string;
  isNovel: boolean;
  category: string;
  baseInputPrice: number;
  baseOutputPrice: number;
  effectiveMarkup: number;
  userInputPrice: number;
  userOutputPrice: number;
  hasOverride: boolean;
}

export default function ModelPricingPage() {
```

```tsx
const queryClient = useQueryClient();
const [selectedModel, setSelectedModel] = useState<ModelPricing | null>(null);
const [bulkMarkup, setBulkMarkup] = useState({ external: 40, selfHosted: 75 });

// Fetch pricing config
const { data: config, isLoading: configLoading } = useQuery<PricingConfig>({
  queryKey: ['pricing-config'],
  queryFn: () => fetch('/api/admin/pricing/config').then(r => r.json()),
});

// Fetch all model pricing
const { data: models, isLoading: modelsLoading } = useQuery<ModelPricing[]>({
  queryKey: ['model-pricing'],
  queryFn: () => fetch('/api/admin/pricing/models').then(r => r.json()),
});

// Update config mutation
const updateConfigMutation = useMutation({
  mutationFn: (data: Partial<PricingConfig>) =>
    fetch('/api/admin/pricing/config', {
      method: 'PUT',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(data),
    }).then(r => r.json()),
  onSuccess: () => {
    queryClient.invalidateQueries({ queryKey: ['pricing-config'] });
    toast.success('Pricing configuration updated');
  },
});

// Bulk update markup mutation
const bulkUpdateMutation = useMutation({
  mutationFn: (data: { type: 'external' | 'self_hosted'; markup: number }) =>
    fetch('/api/admin/pricing/bulk-update', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(data),
    }).then(r => r.json()),
  onSuccess: (_, variables) => {
    queryClient.invalidateQueries({ queryKey: ['model-pricing'] });
    toast.success(`All ${variables.type === 'external' ? 'external' : 'self-hosted'} model
  },
});

// Individual model override mutation
const overrideMutation = useMutation({
```

```
  mutationFn: (data: { modelId: string; markup?: number; inputPrice?: number; outputPrice?
    fetch(`/api/admin/pricing/models/${data.modelId}/override`, {
      method: 'PUT',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(data),
    }).then(r => r.json()),
  onSuccess: () => {
    queryClient.invalidateQueries({ queryKey: ['model-pricing'] });
    setSelectedModel(null);
    toast.success('Model pricing override saved');
  },
});

// Clear override mutation
const clearOverrideMutation = useMutation({
  mutationFn: (modelId: string) =>
    fetch(`/api/admin/pricing/models/${modelId}/override`, {
      method: 'DELETE',
    }).then(r => r.json()),
  onSuccess: () => {
    queryClient.invalidateQueries({ queryKey: ['model-pricing'] });
    toast.success('Pricing override removed');
  },
});

const [localConfig, setLocalConfig] = useState<Partial<PricingConfig>>(config || {});

return (
  <div className="space-y-6">
    <div className="flex justify-between items-center">
      <div>
        <h1 className="text-2xl font-bold">Model Pricing</h1>
        <p className="text-muted-foreground">
          Configure pricing markups and overrides for all AI models
        </p>
      </div>
      <div className="flex gap-2">
        <Button variant="outline" onClick={() => queryClient.invalidateQueries()}>
          <RefreshCw className="h-4 w-4 mr-2" />
          Refresh
        </Button>
        <Button onClick={() => updateConfigMutation.mutate(localConfig)}>
          <Save className="h-4 w-4 mr-2" />
          Save Config
        </Button>
      </div>
```

```
</div>

<Tabs defaultValue="config">
  <TabsList>
    <TabsTrigger value="config">Global Configuration</TabsTrigger>
    <TabsTrigger value="bulk">Bulk Updates</TabsTrigger>
    <TabsTrigger value="models">Individual Models</TabsTrigger>
    <TabsTrigger value="history">Price History</TabsTrigger>
  </TabsList>

  {/* Global Configuration Tab */}
  <TabsContent value="config" className="space-y-4">
    <div className="grid gap-4 md:grid-cols-2">
      <Card>
        <CardHeader>
          <CardTitle className="flex items-center gap-2">
            <Percent className="h-5 w-5" />
            Default Markups
          </CardTitle>
          <CardDescription>
            Global markup percentages applied to all models
          </CardDescription>
        </CardHeader>
        <CardContent className="space-y-6">
          <div className="space-y-3">
            <div className="flex justify-between">
              <Label>External Provider Markup</Label>
              <span className="font-mono text-sm">
                {((localConfig.externalDefaultMarkup || config?.externalDefaultMarkup
              </span>
            </div>
            <Slider
              value={[(localConfig.externalDefaultMarkup || config?.externalDefaultMar
              min={0}
              max={200}
              step={5}
              onValueChange={([value]) =>
                setLocalConfig({ ...localConfig, externalDefaultMarkup: value / 100 })
              }
            />
            <p className="text-xs text-muted-foreground">
              Applied to OpenAI, Anthropic, Google, xAI, Mistral, etc.
            </p>
          </div>

          <div className="space-y-3">
```

```
        <div className="flex justify-between">
          <Label>Self-Hosted Model Markup</Label>
          <span className="font-mono text-sm">
            {((localConfig.selfHostedDefaultMarkup || config?.selfHostedDefaultMar
          </span>
        </div>
        <Slider
          value={[(localConfig.selfHostedDefaultMarkup || config?.selfHostedDefaul
          min={0}
          max={300}
          step={5}
          onValueChange={([value]) =>
            setLocalConfig({ ...localConfig, selfHostedDefaultMarkup: value / 100
          }
        />
        <p className="text-xs text-muted-foreground">
          Applied to SageMaker-hosted models (covers compute costs)
        </p>
      </div>
    </CardContent>
  </Card>

  <Card>
    <CardHeader>
      <CardTitle className="flex items-center gap-2">
        <DollarSign className="h-5 w-5" />
        Pricing Rules
      </CardTitle>
      <CardDescription>
        Additional pricing configuration
      </CardDescription>
    </CardHeader>
    <CardContent className="space-y-4">
      <div className="space-y-2">
        <Label>Minimum Charge Per Request ($)</Label>
        <Input
          type="number"
          step="0.0001"
          value={localConfig.minimumChargePerRequest || config?.minimumChargePerRe
          onChange={(e) =>
            setLocalConfig({ ...localConfig, minimumChargePerRequest: parseFloat(e
          }
        />
      </div>

      <div className="space-y-2">
```

```
            <Label>Price Increase Grace Period (hours)</Label>
            <Input
              type="number"
              min={0}
              max={168}
              value={localConfig.priceIncreaseGracePeriodHours || config?.priceIncreas
              onChange={(e) =>
                setLocalConfig({ ...localConfig, priceIncreaseGracePeriodHours: parse
              }
            />
            <p className="text-xs text-muted-foreground">
              Delay before price increases take effect
            </p>
          </div>

          <div className="flex items-center justify-between">
            <div>
              <Label>Auto-Update from Providers</Label>
              <p className="text-xs text-muted-foreground">
                Automatically sync base prices from provider APIs
              </p>
            </div>
            <Switch
              checked={localConfig.autoUpdateFromProviders ?? config?.autoUpdateFromPr
              onCheckedChange={(checked) =>
                setLocalConfig({ ...localConfig, autoUpdateFromProviders: checked })
              }
            />
          </div>

          <div className="flex items-center justify-between">
            <div>
              <Label>Notify on Price Changes</Label>
              <p className="text-xs text-muted-foreground">
                Alert when provider prices change significantly
              </p>
            </div>
            <Switch
              checked={localConfig.notifyOnPriceChange ?? config?.notifyOnPriceChange
              onCheckedChange={(checked) =>
                setLocalConfig({ ...localConfig, notifyOnPriceChange: checked })
              }
            />
          </div>
        </CardContent>
      </Card>
```

```
  </div>
</TabsContent>

{/* Bulk Updates Tab */}
<TabsContent value="bulk" className="space-y-4">
  <div className="grid gap-4 md:grid-cols-2">
    <Card>
      <CardHeader>
        <CardTitle>External Providers</CardTitle>
        <CardDescription>
          Update markup for all external AI providers at once
        </CardDescription>
      </CardHeader>
      <CardContent className="space-y-4">
        <div className="space-y-3">
          <div className="flex justify-between">
            <Label>Markup Percentage</Label>
            <span className="font-mono text-sm">{bulkMarkup.external}%</span>
          </div>
          <Slider
            value={[bulkMarkup.external]}
            min={0}
            max={200}
            step={5}
            onValueChange={([value]) => setBulkMarkup({ ...bulkMarkup, external: val
          />
        </div>
        <Button
          className="w-full"
          onClick={() => bulkUpdateMutation.mutate({ type: 'external', markup: bulkM
          disabled={bulkUpdateMutation.isPending}
        >
          Apply to All External Models
        </Button>
        <p className="text-xs text-muted-foreground">
          Affects: OpenAI, Anthropic, Google, xAI, Mistral, Perplexity, DeepSeek, Co
        </p>
      </CardContent>
    </Card>

    <Card>
      <CardHeader>
        <CardTitle>Self-Hosted Models</CardTitle>
        <CardDescription>
          Update markup for all SageMaker-hosted models at once
        </CardDescription>
```

```
          </CardHeader>
          <CardContent className="space-y-4">
            <div className="space-y-3">
              <div className="flex justify-between">
                <Label>Markup Percentage</Label>
                <span className="font-mono text-sm">{bulkMarkup.selfHosted}%</span>
              </div>
              <Slider
                value={[bulkMarkup.selfHosted]}
                min={0}
                max={300}
                step={5}
                onValueChange={([value]) => setBulkMarkup({ ...bulkMarkup, selfHosted: v
              />
            </div>
            <Button
              className="w-full"
              onClick={() => bulkUpdateMutation.mutate({ type: 'self_hosted', markup: bu
              disabled={bulkUpdateMutation.isPending}
            >
              Apply to All Self-Hosted Models
            </Button>
            <p className="text-xs text-muted-foreground">
              Affects: Stable Diffusion, Whisper, SAM 2, YOLO, MusicGen, etc.
            </p>
          </CardContent>
        </Card>
      </div>
    </TabsContent>

    {/* Individual Models Tab */}
    <TabsContent value="models">
      <Card>
        <CardHeader>
          <CardTitle>Model Pricing Details</CardTitle>
          <CardDescription>
            View and override pricing for individual models
          </CardDescription>
        </CardHeader>
        <CardContent>
          <Table>
            <TableHeader>
              <TableRow>
                <TableHead>Model</TableHead>
                <TableHead>Provider</TableHead>
                <TableHead>Category</TableHead>
```

```
            <TableHead className="text-right">Base Input</TableHead>
            <TableHead className="text-right">Base Output</TableHead>
            <TableHead className="text-right">Markup</TableHead>
            <TableHead className="text-right">User Input</TableHead>
            <TableHead className="text-right">User Output</TableHead>
            <TableHead>Override</TableHead>
            <TableHead></TableHead>
          </TableRow>
        </TableHeader>
        <TableBody>
          {(models || []).map((model) => (
            <TableRow key={model.modelId}>
              <TableCell>
                <div className="font-medium">{model.displayName}</div>
                <div className="text-xs text-muted-foreground font-mono">{model.mode
              </TableCell>
              <TableCell>{model.providerId}</TableCell>
              <TableCell>
                <Badge variant={model.isNovel ? 'secondary' : 'outline'}>
                  {model.isNovel ? 'Novel' : 'Standard'}
                </Badge>
              </TableCell>
              <TableCell className="text-right font-mono">
                ${model.baseInputPrice.toFixed(2)}
              </TableCell>
              <TableCell className="text-right font-mono">
                ${model.baseOutputPrice.toFixed(2)}
              </TableCell>
              <TableCell className="text-right font-mono">
                {(model.effectiveMarkup * 100).toFixed(0)}%
              </TableCell>
              <TableCell className="text-right font-mono text-green-600">
                ${model.userInputPrice.toFixed(2)}
              </TableCell>
              <TableCell className="text-right font-mono text-green-600">
                ${model.userOutputPrice.toFixed(2)}
              </TableCell>
              <TableCell>
                {model.hasOverride ? (
                  <Badge variant="default">Custom</Badge>
                ) : (
                  <Badge variant="outline">Default</Badge>
                )}
              </TableCell>
              <TableCell>
                <Dialog>
```

```
                          <DialogTrigger asChild>
                            <Button variant="ghost" size="sm" onClick={() => setSelectedMode
                              <Edit2 className="h-4 w-4" />
                            </Button>
                          </DialogTrigger>
                          <DialogContent>
                            <DialogHeader>
                              <DialogTitle>Edit Pricing: {model.displayName}</DialogTitle>
                            </DialogHeader>
                            <ModelPricingEditor
                              model={model}
                              onSave={(data) => overrideMutation.mutate({ modelId: model.mod
                              onClear={() => clearOverrideMutation.mutate(model.modelId)}
                            />
                          </DialogContent>
                        </Dialog>
                      </TableCell>
                    </TableRow>
                  ))}
                </TableBody>
              </Table>
            </CardContent>
          </Card>
        </TabsContent>

        {/* Price History Tab */}
        <TabsContent value="history">
          <PriceHistoryTable />
        </TabsContent>
      </Tabs>
    </div>
  );
}

// Model Pricing Editor Component
function ModelPricingEditor({
  model,
  onSave,
  onClear
}: {
  model: ModelPricing;
  onSave: (data: any) => void;
  onClear: () => void;
}) {
  const [markup, setMarkup] = useState(model.effectiveMarkup * 100);
  const [inputPrice, setInputPrice] = useState<number | null>(null);
```

```
  const [outputPrice, setOutputPrice] = useState<number | null>(null);

  return (
    <div className="space-y-4">
      <div className="space-y-2">
        <Label>Custom Markup (%)</Label>
        <div className="flex items-center gap-4">
          <Slider
            value={[markup]}
            min={0}
            max={200}
            step={5}
            onValueChange={([value]) => setMarkup(value)}
            className="flex-1"
          />
          <span className="font-mono w-16 text-right">{markup}%</span>
        </div>
      </div>

      <div className="grid grid-cols-2 gap-4">
        <div className="space-y-2">
          <Label>Override Input Price ($/1M tokens)</Label>
          <Input
            type="number"
            step="0.01"
            placeholder={`Default: $${model.baseInputPrice.toFixed(2)}`}
            value={inputPrice ?? ''}
            onChange={(e) => setInputPrice(e.target.value ? parseFloat(e.target.value) : nul
          />
        </div>
        <div className="space-y-2">
          <Label>Override Output Price ($/1M tokens)</Label>
          <Input
            type="number"
            step="0.01"
            placeholder={`Default: $${model.baseOutputPrice.toFixed(2)}`}
            value={outputPrice ?? ''}
            onChange={(e) => setOutputPrice(e.target.value ? parseFloat(e.target.value) : nu
          />
        </div>
      </div>

      <div className="bg-muted p-3 rounded-lg">
        <div className="text-sm font-medium mb-2">Preview User Prices</div>
        <div className="grid grid-cols-2 gap-4 text-sm">
          <div>
```

```
              Input: <span className="font-mono text-green-600">
                ${((inputPrice ?? model.baseInputPrice) * (1 + markup / 100)).toFixed(2)}/1M
              </span>
            </div>
            <div>
              Output: <span className="font-mono text-green-600">
                ${((outputPrice ?? model.baseOutputPrice) * (1 + markup / 100)).toFixed(2)}/1M
              </span>
            </div>
          </div>
        </div>

        <DialogFooter className="flex justify-between">
          {model.hasOverride && (
            <Button variant="outline" onClick={onClear}>
              Clear Override
            </Button>
          )}
          <Button onClick={() => onSave({ markup: markup / 100, inputPrice, outputPrice })}>
            Save Override
          </Button>
        </DialogFooter>
      </div>
    );
}

// Price History Table Component
function PriceHistoryTable() {
  const { data: history } = useQuery({
    queryKey: ['price-history'],
    queryFn: () => fetch('/api/admin/pricing/history?limit=100').then(r => r.json()),
  });

  return (
    <Card>
      <CardHeader>
        <CardTitle className="flex items-center gap-2">
          <History className="h-5 w-5" />
          Price Change History
        </CardTitle>
      </CardHeader>
      <CardContent>
        <Table>
          <TableHeader>
            <TableRow>
              <TableHead>Date</TableHead>
```

```
                <TableHead>Model</TableHead>
                <TableHead>Change Type</TableHead>
                <TableHead className="text-right">Previous</TableHead>
                <TableHead className="text-right">New</TableHead>
                <TableHead>Changed By</TableHead>
              </TableRow>
            </TableHeader>
            <TableBody>
              {(history || []).map((entry: any) => (
                <TableRow key={entry.id}>
                  <TableCell className="text-sm">
                    {new Date(entry.createdAt).toLocaleString()}
                  </TableCell>
                  <TableCell className="font-mono text-sm">{entry.modelId}</TableCell>
                  <TableCell>
                    <Badge variant="outline">{entry.changeSource}</Badge>
                  </TableCell>
                  <TableCell className="text-right font-mono text-sm">
                    {entry.previousMarkup ? `${(entry.previousMarkup * 100).toFixed(0)}%` : '-
                  </TableCell>
                  <TableCell className="text-right font-mono text-sm">
                    {entry.newMarkup ? `${(entry.newMarkup * 100).toFixed(0)}%` : '-'}
                  </TableCell>
                  <TableCell className="text-sm">{entry.changedByEmail || 'System'}</TableCell
                </TableRow>
              ))}
            </TableBody>
          </Table>
        </CardContent>
      </Card>
    );
}
```

31.5 Think Tank Model Selection UI

```
// apps/thinktank/src/components/chat/model-selector.tsx

'use client';

import { useState, useMemo } from 'react';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import {
  Popover, PopoverContent, PopoverTrigger
} from '@/components/ui/popover';
```

```tsx
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Badge } from '@/components/ui/badge';
import { Switch } from '@/components/ui/switch';
import { Label } from '@/components/ui/label';
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';
import { ScrollArea } from '@/components/ui/scroll-area';
import { cn } from '@/lib/utils';
import {
  ChevronDown, Search, Star, StarOff, Sparkles, Zap,
  DollarSign, Brain, Check, Settings2
} from 'lucide-react';

interface Model {
  id: string;
  displayName: string;
  providerId: string;
  providerName: string;
  isNovel: boolean;
  category: string;
  contextWindow: number;
  capabilities: string[];
  userInputPrice: number;
  userOutputPrice: number;
  isFavorite?: boolean;
}

interface ModelPreferences {
  selectionMode: 'auto' | 'manual' | 'favorites';
  defaultModelId?: string;
  favoriteModels: string[];
  showStandardModels: boolean;
  showNovelModels: boolean;
  showCostPerMessage: boolean;
  maxCostPerMessage?: number;
}

interface ModelSelectorProps {
  selectedModel: string | null; // null = Auto
  onModelChange: (modelId: string | null) => void;
  disabled?: boolean;
}

export function ModelSelector({ selectedModel, onModelChange, disabled }: ModelSelectorProps
  const [open, setOpen] = useState(false);
  const [search, setSearch] = useState('');
```

```tsx
const queryClient = useQueryClient();

// Fetch available models
const { data: models = [] } = useQuery<Model[]>({
  queryKey: ['thinktank-models'],
  queryFn: () => fetch('/api/thinktank/models').then(r => r.json()),
});

// Fetch user preferences
const { data: preferences } = useQuery<ModelPreferences>({
  queryKey: ['thinktank-model-preferences'],
  queryFn: () => fetch('/api/thinktank/preferences/models').then(r => r.json()),
});

// Toggle favorite mutation
const toggleFavoriteMutation = useMutation({
  mutationFn: (modelId: string) =>
    fetch('/api/thinktank/preferences/models/favorite', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ modelId }),
    }).then(r => r.json()),
  onSuccess: () => {
    queryClient.invalidateQueries({ queryKey: ['thinktank-model-preferences'] });
    queryClient.invalidateQueries({ queryKey: ['thinktank-models'] });
  },
});

// Filter and group models
const { standardModels, novelModels, favoriteModels } = useMemo(() => {
  const filtered = models.filter(m =>
    m.displayName.toLowerCase().includes(search.toLowerCase()) ||
    m.providerId.toLowerCase().includes(search.toLowerCase())
  );

  return {
    standardModels: filtered.filter(m => !m.isNovel && preferences?.showStandardModels !==
    novelModels: filtered.filter(m => m.isNovel && preferences?.showNovelModels !== false)
    favoriteModels: filtered.filter(m => preferences?.favoriteModels?.includes(m.id)),
  };
}, [models, search, preferences]);

// Get selected model details
const selectedModelDetails = selectedModel
  ? models.find(m => m.id === selectedModel)
  : null;
```

```
// Format price for display
const formatPrice = (inputPrice: number, outputPrice: number) => {
  const avgPrice = (inputPrice + outputPrice) / 2;
  if (avgPrice < 1) return `$${avgPrice.toFixed(3)}/1K`;
  return `$${avgPrice.toFixed(2)}/1M`;
};

return (
  <Popover open={open} onOpenChange={setOpen}>
    <PopoverTrigger asChild>
      <Button
        variant="outline"
        role="combobox"
        aria-expanded={open}
        disabled={disabled}
        className="justify-between min-w-[200px]"
      >
        <div className="flex items-center gap-2">
          {selectedModel === null ? (
            <>
              <Brain className="h-4 w-4 text-purple-500" />
              <span>Auto</span>
              <Badge variant="secondary" className="text-xs">RADIANT Brain</Badge>
            </>
          ) : (
            <>
              {selectedModelDetails?.isNovel && <Sparkles className="h-4 w-4 text-amber-50
              <span>{selectedModelDetails?.displayName || selectedModel}</span>
              {preferences?.showCostPerMessage && selectedModelDetails && (
                <span className="text-xs text-muted-foreground">
                  {formatPrice(selectedModelDetails.userInputPrice, selectedModelDetails.u
                </span>
              )}
            </>
          )}
        </div>
        <ChevronDown className="ml-2 h-4 w-4 shrink-0 opacity-50" />
      </Button>
    </PopoverTrigger>

    <PopoverContent className="w-[400px] p-0" align="start">
      <div className="p-3 border-b">
        <div className="relative">
          <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-muted-f
          <Input
```

```
      placeholder="Search models..."
      value={search}
      onChange={(e) => setSearch(e.target.value)}
      className="pl-9"
    />
  </div>
</div>

<Tabs defaultValue="all" className="w-full">
  <TabsList className="w-full justify-start rounded-none border-b bg-transparent p-0
    <TabsTrigger value="all" className="rounded-none data-[state=active]:border-b-2"
      All
    </TabsTrigger>
    <TabsTrigger value="favorites" className="rounded-none data-[state=active]:borde
      <Star className="h-3 w-3 mr-1" />
      Favorites
    </TabsTrigger>
    <TabsTrigger value="standard" className="rounded-none data-[state=active]:borde
      Standard
    </TabsTrigger>
    <TabsTrigger value="novel" className="rounded-none data-[state=active]:border-b-
      <Sparkles className="h-3 w-3 mr-1" />
      Novel
    </TabsTrigger>
  </TabsList>

  <ScrollArea className="h-[300px]">
    {/* Auto Option */}
    <div className="p-1">
      <ModelOption
        model={null}
        isSelected={selectedModel === null}
        onSelect={() => {
          onModelChange(null);
          setOpen(false);
        }}
        showCost={false}
      />
    </div>

    <TabsContent value="all" className="m-0 p-1">
      {favoriteModels.length > 0 && (
        <div className="mb-2">
          <div className="px-2 py-1 text-xs font-medium text-muted-foreground">Favor
          {favoriteModels.map(model => (
            <ModelOption
```

```
            key={model.id}
            model={model}
            isSelected={selectedModel === model.id}
            isFavorite={true}
            onSelect={() => {
              onModelChange(model.id);
              setOpen(false);
            }}
            onToggleFavorite={() => toggleFavoriteMutation.mutate(model.id)}
            showCost={preferences?.showCostPerMessage}
          />
        ))}
      </div>
    )}

    {standardModels.length > 0 && (
      <div className="mb-2">
        <div className="px-2 py-1 text-xs font-medium text-muted-foreground">Stand
        {standardModels.map(model => (
          <ModelOption
            key={model.id}
            model={model}
            isSelected={selectedModel === model.id}
            isFavorite={preferences?.favoriteModels?.includes(model.id)}
            onSelect={() => {
              onModelChange(model.id);
              setOpen(false);
            }}
            onToggleFavorite={() => toggleFavoriteMutation.mutate(model.id)}
            showCost={preferences?.showCostPerMessage}
          />
        ))}
      </div>
    )}

    {novelModels.length > 0 && (
      <div>
        <div className="px-2 py-1 text-xs font-medium text-muted-foreground flex i
          <Sparkles className="h-3 w-3" />
          Novel / Experimental
        </div>
        {novelModels.map(model => (
          <ModelOption
            key={model.id}
            model={model}
            isSelected={selectedModel === model.id}
```

```
            isFavorite={preferences?.favoriteModels?.includes(model.id)}
            onSelect={() => {
              onModelChange(model.id);
              setOpen(false);
            }}
            onToggleFavorite={() => toggleFavoriteMutation.mutate(model.id)}
            showCost={preferences?.showCostPerMessage}
          />
        ))}
      </div>
    )}
  </TabsContent>

  <TabsContent value="favorites" className="m-0 p-1">
    {favoriteModels.length === 0 ? (
      <div className="p-4 text-center text-sm text-muted-foreground">
        No favorite models yet. Star models to add them here.
      </div>
    ) : (
      favoriteModels.map(model => (
        <ModelOption
          key={model.id}
          model={model}
          isSelected={selectedModel === model.id}
          isFavorite={true}
          onSelect={() => {
            onModelChange(model.id);
            setOpen(false);
          }}
          onToggleFavorite={() => toggleFavoriteMutation.mutate(model.id)}
          showCost={preferences?.showCostPerMessage}
        />
      ))
    )}
  </TabsContent>

  <TabsContent value="standard" className="m-0 p-1">
    {standardModels.map(model => (
      <ModelOption
        key={model.id}
        model={model}
        isSelected={selectedModel === model.id}
        isFavorite={preferences?.favoriteModels?.includes(model.id)}
        onSelect={() => {
          onModelChange(model.id);
          setOpen(false);
```

```
                    }}
                    onToggleFavorite={() => toggleFavoriteMutation.mutate(model.id)}
                    showCost={preferences?.showCostPerMessage}
                  />
                ))}
              </TabsContent>

              <TabsContent value="novel" className="m-0 p-1">
                {novelModels.map(model => (
                  <ModelOption
                    key={model.id}
                    model={model}
                    isSelected={selectedModel === model.id}
                    isFavorite={preferences?.favoriteModels?.includes(model.id)}
                    onSelect={() => {
                      onModelChange(model.id);
                      setOpen(false);
                    }}
                    onToggleFavorite={() => toggleFavoriteMutation.mutate(model.id)}
                    showCost={preferences?.showCostPerMessage}
                  />
                ))}
              </TabsContent>
            </ScrollArea>
          </Tabs>

          {/* Settings Footer */}
          <div className="border-t p-2 flex justify-between items-center">
            <div className="flex items-center gap-2 text-xs text-muted-foreground">
              <DollarSign className="h-3 w-3" />
              <span>Prices per 1M tokens</span>
            </div>
            <Button variant="ghost" size="sm" className="text-xs">
              <Settings2 className="h-3 w-3 mr-1" />
              Preferences
            </Button>
          </div>
        </PopoverContent>
      </Popover>
    );
}

// Individual Model Option Component
function ModelOption({
  model,
  isSelected,
```

```
  isFavorite,
  onSelect,
  onToggleFavorite,
  showCost,
}: {
  model: Model | null;
  isSelected: boolean;
  isFavorite?: boolean;
  onSelect: () => void;
  onToggleFavorite?: () => void;
  showCost?: boolean;
}) {
  // Auto option
  if (model === null) {
    return (
      <div
        className={cn(
          "flex items-center gap-3 p-2 rounded-md cursor-pointer hover:bg-accent",
          isSelected && "bg-accent"
        )}
        onClick={onSelect}
      >
        <Brain className="h-5 w-5 text-purple-500" />
        <div className="flex-1">
          <div className="flex items-center gap-2">
            <span className="font-medium">Auto</span>
            <Badge variant="secondary" className="text-xs">RADIANT Brain</Badge>
          </div>
          <div className="text-xs text-muted-foreground">
            Intelligently selects the best model for your task
          </div>
        </div>
        {isSelected && <Check className="h-4 w-4 text-primary" />}
      </div>
    );
  }

  return (
    <div
      className={cn(
        "flex items-center gap-3 p-2 rounded-md cursor-pointer hover:bg-accent group",
        isSelected && "bg-accent"
      )}
      onClick={onSelect}
    >
      <div className="flex-shrink-0">
```

```
  {model.isNovel ? (
    <Sparkles className="h-5 w-5 text-amber-500" />
  ) : (
    <Zap className="h-5 w-5 text-blue-500" />
  )}
</div>

<div className="flex-1 min-w-0">
  <div className="flex items-center gap-2">
    <span className="font-medium truncate">{model.displayName}</span>
    {model.isNovel && (
      <Badge variant="outline" className="text-xs text-amber-600 border-amber-300">
        Novel
      </Badge>
    )}
  </div>
  <div className="flex items-center gap-2 text-xs text-muted-foreground">
    <span>{model.providerName}</span>
    <span>•</span>
    <span>{(model.contextWindow / 1000).toFixed(0)}K context</span>
    {showCost && (
      <>
        <span>•</span>
        <span className="text-green-600">
          ${((model.userInputPrice + model.userOutputPrice) / 2).toFixed(2)}/1M
        </span>
      </>
    )}
  </div>
</div>

<div className="flex items-center gap-1">
  {onToggleFavorite && (
    <button
      className="p-1 opacity-0 group-hover:opacity-100 transition-opacity"
      onClick={(e) => {
        e.stopPropagation();
        onToggleFavorite();
      }}
    >
      {isFavorite ? (
        <Star className="h-4 w-4 text-yellow-500 fill-yellow-500" />
      ) : (
        <StarOff className="h-4 w-4 text-muted-foreground" />
      )}
    </button>
```

```
        )}
        {isSelected && <Check className="h-4 w-4 text-primary" />}
      </div>
    </div>
  );
}
```

31.6 Think Tank Model API Endpoints

```
// packages/lambdas/src/handlers/thinktank/models.ts

import { APIGatewayProxyEvent, APIGatewayProxyResult } from 'aws-lambda';
import { Pool } from 'pg';
import { createLogger, corsHeaders } from '@radiant/shared';
import { verifyJWT, getUserFromToken } from '../auth/jwt';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });
const logger = createLogger('thinktank-models');

// GET /api/thinktank/models - List available models for Think Tank users
export async function listModels(event: APIGatewayProxyEvent): Promise<APIGatewayProxyResult
  try {
    const user = await getUserFromToken(event);
    const client = await pool.connect();

    try {
      // Get models with effective pricing and user favorites
      const result = await client.query(`
        SELECT
          m.id,
          m.display_name,
          m.provider_id,
          p.display_name as provider_name,
          m.is_novel,
          m.category,
          m.context_window,
          m.capabilities,
          m.thinktank_enabled,

          -- Effective pricing (from view)
          emp.user_input_price,
          emp.user_output_price,
          emp.effective_markup,
```

```
        -- User favorites
        $1 = ANY(COALESCE(ump.favorite_models, '[]')::text[]) as is_favorite

      FROM models m
      JOIN providers p ON m.provider_id = p.id
      LEFT JOIN effective_model_pricing emp ON m.id = emp.model_id
      LEFT JOIN thinktank_user_model_preferences ump ON ump.user_id = $2

      WHERE m.thinktank_enabled = true
        AND m.is_enabled = true
        AND m.status = 'active'

      ORDER BY
        m.is_novel ASC,
        m.thinktank_display_order ASC,
        m.display_name ASC
    `, [user.id, user.id]);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify(result.rows.map(row => ({
        id: row.id,
        displayName: row.display_name,
        providerId: row.provider_id,
        providerName: row.provider_name,
        isNovel: row.is_novel,
        category: row.category,
        contextWindow: row.context_window,
        capabilities: row.capabilities || [],
        userInputPrice: parseFloat(row.user_input_price) || 0,
        userOutputPrice: parseFloat(row.user_output_price) || 0,
        isFavorite: row.is_favorite,
      }))),
    };
  } finally {
    client.release();
  }
} catch (error) {
  logger.error('Failed to list models', { error });
  return {
    statusCode: 500,
    headers: corsHeaders,
    body: JSON.stringify({ error: 'Failed to load models' }),
  };
}
```

```
}

// GET /api/thinktank/preferences/models - Get user's model preferences
export async function getModelPreferences(event: APIGatewayProxyEvent): Promise<APIGatewayPr
  try {
    const user = await getUserFromToken(event);
    const client = await pool.connect();

    try {
      const result = await client.query(`
        SELECT
          selection_mode,
          default_model_id,
          favorite_models,
          show_standard_models,
          show_novel_models,
          show_self_hosted_models,
          show_cost_per_message,
          max_cost_per_message,
          prefer_cost_optimization,
          domain_mode_model_overrides,
          recent_models
        FROM thinktank_user_model_preferences
        WHERE user_id = $1
      `, [user.id]);

      if (result.rows.length === 0) {
        // Return defaults
        return {
          statusCode: 200,
          headers: corsHeaders,
          body: JSON.stringify({
            selectionMode: 'auto',
            defaultModelId: null,
            favoriteModels: [],
            showStandardModels: true,
            showNovelModels: true,
            showSelfHostedModels: false,
            showCostPerMessage: true,
            maxCostPerMessage: null,
            preferCostOptimization: false,
            domainModeModelOverrides: {},
            recentModels: [],
          }),
        };
      }
```

```
      const row = result.rows[0];
      return {
        statusCode: 200,
        headers: corsHeaders,
        body: JSON.stringify({
          selectionMode: row.selection_mode,
          defaultModelId: row.default_model_id,
          favoriteModels: row.favorite_models || [],
          showStandardModels: row.show_standard_models,
          showNovelModels: row.show_novel_models,
          showSelfHostedModels: row.show_self_hosted_models,
          showCostPerMessage: row.show_cost_per_message,
          maxCostPerMessage: row.max_cost_per_message ? parseFloat(row.max_cost_per_message)
          preferCostOptimization: row.prefer_cost_optimization,
          domainModeModelOverrides: row.domain_mode_model_overrides || {},
          recentModels: row.recent_models || [],
        }),
      };
    } finally {
      client.release();
    }
  } catch (error) {
    logger.error('Failed to get model preferences', { error });
    return {
      statusCode: 500,
      headers: corsHeaders,
      body: JSON.stringify({ error: 'Failed to load preferences' }),
    };
  }
}

// PUT /api/thinktank/preferences/models - Update user's model preferences
export async function updateModelPreferences(event: APIGatewayProxyEvent): Promise<APIGatewa
  try {
    const user = await getUserFromToken(event);
    const body = JSON.parse(event.body || '{}');
    const client = await pool.connect();

    try {
      await client.query(`
        INSERT INTO thinktank_user_model_preferences (
          user_id, tenant_id, selection_mode, default_model_id, favorite_models,
          show_standard_models, show_novel_models, show_self_hosted_models,
          show_cost_per_message, max_cost_per_message, prefer_cost_optimization,
          domain_mode_model_overrides
```

```
          ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12)
          ON CONFLICT (user_id) DO UPDATE SET
            selection_mode = COALESCE($3, thinktank_user_model_preferences.selection_mode),
            default_model_id = COALESCE($4, thinktank_user_model_preferences.default_model_id)
            favorite_models = COALESCE($5, thinktank_user_model_preferences.favorite_models),
            show_standard_models = COALESCE($6, thinktank_user_model_preferences.show_standard
            show_novel_models = COALESCE($7, thinktank_user_model_preferences.show_novel_model
            show_self_hosted_models = COALESCE($8, thinktank_user_model_preferences.show_self_
            show_cost_per_message = COALESCE($9, thinktank_user_model_preferences.show_cost_pe
            max_cost_per_message = $10,
            prefer_cost_optimization = COALESCE($11, thinktank_user_model_preferences.prefer_c
            domain_mode_model_overrides = COALESCE($12, thinktank_user_model_preferences.domai
            updated_at = NOW()
        `, [
          user.id,
          user.tenantId,
          body.selectionMode,
          body.defaultModelId,
          JSON.stringify(body.favoriteModels || []),
          body.showStandardModels,
          body.showNovelModels,
          body.showSelfHostedModels,
          body.showCostPerMessage,
          body.maxCostPerMessage,
          body.preferCostOptimization,
          JSON.stringify(body.domainModeModelOverrides || {}),
        ]);

        return {
          statusCode: 200,
          headers: corsHeaders,
          body: JSON.stringify({ success: true }),
        };
      } finally {
        client.release();
      }
    } catch (error) {
      logger.error('Failed to update model preferences', { error });
      return {
        statusCode: 500,
        headers: corsHeaders,
        body: JSON.stringify({ error: 'Failed to save preferences' }),
      };
    }
  }
```

```typescript
// POST /api/thinktank/preferences/models/favorite - Toggle favorite model
export async function toggleFavoriteModel(event: APIGatewayProxyEvent): Promise<APIGatewayPr
  try {
    const user = await getUserFromToken(event);
    const { modelId } = JSON.parse(event.body || '{}');

    if (!modelId) {
      return {
        statusCode: 400,
        headers: corsHeaders,
        body: JSON.stringify({ error: 'modelId is required' }),
      };
    }

    const client = await pool.connect();

    try {
      // Check if model is currently a favorite
      const currentResult = await client.query(`
        SELECT favorite_models FROM thinktank_user_model_preferences WHERE user_id = $1
      `, [user.id]);

      let favorites: string[] = currentResult.rows[0]?.favorite_models || [];

      if (favorites.includes(modelId)) {
        // Remove from favorites
        favorites = favorites.filter(id => id !== modelId);
      } else {
        // Add to favorites
        favorites.push(modelId);
      }

      // Update or insert
      await client.query(`
        INSERT INTO thinktank_user_model_preferences (user_id, tenant_id, favorite_models)
        VALUES ($1, $2, $3)
        ON CONFLICT (user_id) DO UPDATE SET
          favorite_models = $3,
          updated_at = NOW()
      `, [user.id, user.tenantId, JSON.stringify(favorites)]);

      return {
        statusCode: 200,
        headers: corsHeaders,
        body: JSON.stringify({
          success: true,
```

```
          isFavorite: favorites.includes(modelId),
          favoriteModels: favorites,
        }),
      };
    } finally {
      client.release();
    }
  } catch (error) {
    logger.error('Failed to toggle favorite', { error });
    return {
      statusCode: 500,
      headers: corsHeaders,
      body: JSON.stringify({ error: 'Failed to update favorites' }),
    };
  }
}
```

31.7 Admin Pricing API Endpoints

```
// packages/lambdas/src/handlers/admin/pricing.ts

import { APIGatewayProxyEvent, APIGatewayProxyResult } from 'aws-lambda';
import { Pool } from 'pg';
import { createLogger, corsHeaders } from '@radiant/shared';
import { requireAdmin } from '../auth/admin';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });
const logger = createLogger('admin-pricing');

// GET /api/admin/pricing/config
export async function getPricingConfig(event: APIGatewayProxyEvent): Promise<APIGatewayProxy
  try {
    await requireAdmin(event);
    const client = await pool.connect();

    try {
      const result = await client.query(`
        SELECT * FROM pricing_config LIMIT 1
      `);

      if (result.rows.length === 0) {
        // Return defaults
        return {
          statusCode: 200,
```

```
            headers: corsHeaders,
            body: JSON.stringify({
              externalDefaultMarkup: 0.40,
              selfHostedDefaultMarkup: 0.75,
              minimumChargePerRequest: 0.001,
              priceIncreaseGracePeriodHours: 24,
              autoUpdateFromProviders: true,
              autoUpdateFrequency: 'daily',
              notifyOnPriceChange: true,
              notifyThresholdPercent: 10,
            }),
          };
        }

        const row = result.rows[0];
        return {
          statusCode: 200,
          headers: corsHeaders,
          body: JSON.stringify({
            externalDefaultMarkup: parseFloat(row.external_default_markup),
            selfHostedDefaultMarkup: parseFloat(row.self_hosted_default_markup),
            minimumChargePerRequest: parseFloat(row.minimum_charge_per_request),
            priceIncreaseGracePeriodHours: row.price_increase_grace_period_hours,
            autoUpdateFromProviders: row.auto_update_from_providers,
            autoUpdateFrequency: row.auto_update_frequency,
            lastAutoUpdate: row.last_auto_update,
            notifyOnPriceChange: row.notify_on_price_change,
            notifyThresholdPercent: parseFloat(row.notify_threshold_percent),
          }),
        };
      } finally {
        client.release();
      }
    } catch (error) {
      logger.error('Failed to get pricing config', { error });
      return {
        statusCode: 500,
        headers: corsHeaders,
        body: JSON.stringify({ error: 'Failed to load pricing config' }),
      };
    }
  }

  // PUT /api/admin/pricing/config
  export async function updatePricingConfig(event: APIGatewayProxyEvent): Promise<APIGatewayPr
    try {
```

```
  const admin = await requireAdmin(event);
  const body = JSON.parse(event.body || '{}');
  const client = await pool.connect();

  try {
    await client.query(`
      INSERT INTO pricing_config (
        tenant_id, external_default_markup, self_hosted_default_markup,
        minimum_charge_per_request, price_increase_grace_period_hours,
        auto_update_from_providers, auto_update_frequency,
        notify_on_price_change, notify_threshold_percent
      ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)
      ON CONFLICT (tenant_id) DO UPDATE SET
        external_default_markup = COALESCE($2, pricing_config.external_default_markup),
        self_hosted_default_markup = COALESCE($3, pricing_config.self_hosted_default_marku
        minimum_charge_per_request = COALESCE($4, pricing_config.minimum_charge_per_reques
        price_increase_grace_period_hours = COALESCE($5, pricing_config.price_increase_gra
        auto_update_from_providers = COALESCE($6, pricing_config.auto_update_from_provider
        auto_update_frequency = COALESCE($7, pricing_config.auto_update_frequency),
        notify_on_price_change = COALESCE($8, pricing_config.notify_on_price_change),
        notify_threshold_percent = COALESCE($9, pricing_config.notify_threshold_percent),
        updated_at = NOW()
    `, [
      admin.tenantId,
      body.externalDefaultMarkup,
      body.selfHostedDefaultMarkup,
      body.minimumChargePerRequest,
      body.priceIncreaseGracePeriodHours,
      body.autoUpdateFromProviders,
      body.autoUpdateFrequency,
      body.notifyOnPriceChange,
      body.notifyThresholdPercent,
    ]);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({ success: true }),
    };
  } finally {
    client.release();
  }
} catch (error) {
  logger.error('Failed to update pricing config', { error });
  return {
    statusCode: 500,
```

```
      headers: corsHeaders,
      body: JSON.stringify({ error: 'Failed to save config' }),
    };
  }
}

// POST /api/admin/pricing/bulk-update - Bulk update markups
export async function bulkUpdatePricing(event: APIGatewayProxyEvent): Promise<APIGatewayProx
  try {
    const admin = await requireAdmin(event);
    const { type, markup } = JSON.parse(event.body || '{}');

    if (!type || markup === undefined) {
      return {
        statusCode: 400,
        headers: corsHeaders,
        body: JSON.stringify({ error: 'type and markup are required' }),
      };
    }

    const client = await pool.connect();

    try {
      await client.query('BEGIN');

      // Get affected models
      const modelsResult = await client.query(`
        SELECT id, pricing->>'billed_markup' as current_markup
        FROM models
        WHERE provider_id ${type === 'self_hosted' ? "= 'self_hosted'" : "!= 'self_hosted'"}
      `);

      // Record price history for each model
      for (const model of modelsResult.rows) {
        await client.query(`
          INSERT INTO price_history (tenant_id, model_id, previous_markup, new_markup, chang
          VALUES ($1, $2, $3, $4, 'bulk_update', $5)
        `, [admin.tenantId, model.id, parseFloat(model.current_markup) || 0, markup / 100, a
      }

      // Update all models of the specified type
      await client.query(`
        UPDATE models
        SET pricing = jsonb_set(
          COALESCE(pricing, '{}'::jsonb),
          '{billed_markup}',
```

```
            to_jsonb($1::numeric)
          ),
          updated_at = NOW()
          WHERE provider_id ${type === 'self_hosted' ? "= 'self_hosted'" : "!= 'self_hosted'"}
        `, [markup / 100]);

        // Also update the default in pricing_config
        if (type === 'self_hosted') {
          await client.query(`
            UPDATE pricing_config SET self_hosted_default_markup = $1, updated_at = NOW()
            WHERE tenant_id = $2
          `, [markup / 100, admin.tenantId]);
        } else {
          await client.query(`
            UPDATE pricing_config SET external_default_markup = $1, updated_at = NOW()
            WHERE tenant_id = $2
          `, [markup / 100, admin.tenantId]);
        }

        await client.query('COMMIT');

        return {
          statusCode: 200,
          headers: corsHeaders,
          body: JSON.stringify({
            success: true,
            modelsUpdated: modelsResult.rows.length,
          }),
        };
      } catch (error) {
        await client.query('ROLLBACK');
        throw error;
      } finally {
        client.release();
      }
    } catch (error) {
      logger.error('Failed to bulk update pricing', { error });
      return {
        statusCode: 500,
        headers: corsHeaders,
        body: JSON.stringify({ error: 'Failed to update pricing' }),
      };
    }
  }

  // PUT /api/admin/pricing/models/:modelId/override - Set individual model override
```

```
export async function setModelPricingOverride(event: APIGatewayProxyEvent): Promise<APIGatew
  try {
    const admin = await requireAdmin(event);
    const modelId = event.pathParameters?.modelId;
    const { markup, inputPrice, outputPrice } = JSON.parse(event.body || '{}');

    if (!modelId) {
      return {
        statusCode: 400,
        headers: corsHeaders,
        body: JSON.stringify({ error: 'modelId is required' }),
      };
    }

    const client = await pool.connect();

    try {
      await client.query('BEGIN');

      // Get current values for history
      const currentResult = await client.query(`
        SELECT markup_override, input_price_override, output_price_override
        FROM model_pricing_overrides
        WHERE tenant_id = $1 AND model_id = $2
        ORDER BY effective_from DESC
        LIMIT 1
      `, [admin.tenantId, modelId]);

      const current = currentResult.rows[0];

      // Record history
      await client.query(`
        INSERT INTO price_history (
          tenant_id, model_id,
          previous_markup, new_markup,
          previous_input_price, new_input_price,
          previous_output_price, new_output_price,
          change_source, changed_by
        ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, 'admin', $9)
      `, [
        admin.tenantId, modelId,
        current?.markup_override, markup,
        current?.input_price_override, inputPrice,
        current?.output_price_override, outputPrice,
        admin.id,
      ]);
```

```
      // Insert or update override
      await client.query(`
        INSERT INTO model_pricing_overrides (
          tenant_id, model_id, markup_override, input_price_override, output_price_override,
        ) VALUES ($1, $2, $3, $4, $5, $6)
        ON CONFLICT (tenant_id, model_id, effective_from) DO UPDATE SET
          markup_override = $3,
          input_price_override = $4,
          output_price_override = $5,
          updated_at = NOW()
      `, [admin.tenantId, modelId, markup, inputPrice, outputPrice, admin.id]);

      await client.query('COMMIT');

      return {
        statusCode: 200,
        headers: corsHeaders,
        body: JSON.stringify({ success: true }),
      };
    } catch (error) {
      await client.query('ROLLBACK');
      throw error;
    } finally {
      client.release();
    }
  } catch (error) {
    logger.error('Failed to set pricing override', { error });
    return {
      statusCode: 500,
      headers: corsHeaders,
      body: JSON.stringify({ error: 'Failed to save override' }),
    };
  }
}

// DELETE /api/admin/pricing/models/:modelId/override - Remove override
export async function deleteModelPricingOverride(event: APIGatewayProxyEvent): Promise<APIGa
  try {
    const admin = await requireAdmin(event);
    const modelId = event.pathParameters?.modelId;

    if (!modelId) {
      return {
        statusCode: 400,
        headers: corsHeaders,
```

```
        body: JSON.stringify({ error: 'modelId is required' }),
      };
    }

    const client = await pool.connect();

    try {
      await client.query(`
        DELETE FROM model_pricing_overrides
        WHERE tenant_id = $1 AND model_id = $2
      `, [admin.tenantId, modelId]);

      return {
        statusCode: 200,
        headers: corsHeaders,
        body: JSON.stringify({ success: true }),
      };
    } finally {
      client.release();
    }
  } catch (error) {
    logger.error('Failed to delete pricing override', { error });
    return {
      statusCode: 500,
      headers: corsHeaders,
      body: JSON.stringify({ error: 'Failed to remove override' }),
    };
  }
}
```

31.8 Integration with Chat Handler

```
// apps/thinktank/src/components/chat/chat-input.tsx (updated)

import { ModelSelector } from './model-selector';

export function ChatInput({ onSend, disabled }: ChatInputProps) {
  const [message, setMessage] = useState('');
  const [selectedModel, setSelectedModel] = useState<string | null>(null); // null = Auto

  const handleSend = () => {
    if (!message.trim()) return;

    onSend({
```

```
      content: message,
      modelId: selectedModel, // Will be resolved by RADIANT Brain if null
    });

    setMessage('');
  };

  return (
    <div className="border-t p-4">
      <div className="flex items-end gap-3">
        <div className="flex-1">
          <Textarea
            value={message}
            onChange={(e) => setMessage(e.target.value)}
            placeholder="Type your message..."
            className="min-h-[60px] resize-none"
            disabled={disabled}
            onKeyDown={(e) => {
              if (e.key === 'Enter' && !e.shiftKey) {
                e.preventDefault();
                handleSend();
              }
            }}
          />
        </div>
        <div className="flex flex-col gap-2">
          <ModelSelector
            selectedModel={selectedModel}
            onModelChange={setSelectedModel}
            disabled={disabled}
          />
          <Button onClick={handleSend} disabled={disabled || !message.trim()}>
            <Send className="h-4 w-4" />
          </Button>
        </div>
      </div>
    </div>
  );
}
```

SECTION-32-TIME-MACHINE-CORE

SECTION 32: TIME MACHINE CORE - DATABASE & SERVICE LAYER (v4.0.0)

Version: 4.0.0 | Apple Time Machine-inspired chat history for Think Tank NEVER lose a chat or file - everything is preserved and recoverable forever

---

32.1 Time Machine Design Philosophy

Inspired by Apple Time Machine's Best Parts

```
APPLE TIME MACHINE INSPIRATION


WHAT WE'RE BORROWING:

1. Visual "fly back through time" - messages recede into the past
2. Calendar-based navigation - pick any date to jump to
3. Timeline bar on the side - scrub to any point
4. One-click restore - instantly recover anything
5. "Enter Time Machine" mode - separate from normal view
6. Everything is automatic - no manual "save" needed
7. Never delete anything - space is cheap, data is priceless

RADIANT IMPROVEMENTS:

1. Works for chat AND files (unified versioning)
2. API-first - client apps can build their own Time Machine UI
3. AI-aware - simplified API lets AI help users navigate history
4. Real-time - see changes as they happen, not just hourly backups
5. Granular - restore single message OR entire conversation
6. Searchable - find that thing you said 3 months ago
```

The Golden Rules

1. **AUTOMATIC** - Every action creates a snapshot. Users never "save."
2. **INVISIBLE** - Hidden until needed. Default UI is just simple chat.

3. **COMPLETE** - Messages, files, edits, metadata - everything versioned.
4. **INSTANT** - Restore happens in milliseconds, not minutes.
5. **FOREVER** - Nothing is ever truly deleted. Soft-delete only.

32.2 Core Types

```
// packages/shared/src/types/time-machine.ts

//
// TIME MACHINE CORE TYPES
//

export type SnapshotTrigger =
  | 'message_sent'        // User sent a message
  | 'message_received'    // AI responded
  | 'message_edited'      // User edited a message
  | 'message_deleted'     // User "deleted" (soft) a message
  | 'file_uploaded'       // User uploaded a file
  | 'file_generated'      // AI generated a file
  | 'file_deleted'        // User "deleted" (soft) a file
  | 'chat_renamed'        // Chat title changed
  | 'restore_performed'   // User restored from history
  | 'manual_snapshot';    // User explicitly saved a point

export type RestoreScope =
  | 'full_chat'           // Restore entire chat to that point
  | 'single_message'      // Restore just one message
  | 'single_file'         // Restore just one file
  | 'message_range'       // Restore a range of messages
  | 'files_only';         // Restore all files, keep messages

export type MediaStatus =
  | 'active'              // Currently visible to user
  | 'processing'          // Being uploaded/processed
  | 'archived'            // Moved to cold storage (still retrievable)
  | 'soft_deleted';       // User "deleted" but still exists

export type ExportFormat = 'zip' | 'json' | 'markdown' | 'pdf' | 'html';

//
// SNAPSHOT - Point in time capture of chat state
//

export interface TimeMachineSnapshot {
```

```
  id: string;
  chatId: string;
  tenantId: string;

  // Version info
  version: number;                    // Monotonically increasing
  timestamp: string;                  // ISO 8601 with milliseconds

  // State summary at this point
  messageCount: number;
  fileCount: number;
  totalTokens: number;

  // What triggered this snapshot
  trigger: SnapshotTrigger;
  triggerDetails?: {
    messageId?: string;
    fileId?: string;
    description?: string;
  };

  // Lineage
  previousSnapshotId?: string;
  restoredFromSnapshotId?: string;    // If this was created by a restore

  // Integrity
  checksum: string;                   // SHA-256 of content

  // Metadata
  createdAt: string;
}

//
// MESSAGE VERSION - Every edit creates a new version
//

export interface MessageVersion {
  id: string;
  messageId: string;                  // Stable ID across versions
  tenantId: string;
  snapshotId: string;

  // Content
  content: string;
  role: 'user' | 'assistant' | 'system';
  modelId?: string;
```

```
  // Version info
  version: number;
  isActive: boolean;                    // Is this the current version?
  isSoftDeleted: boolean;

  // Edit tracking
  editReason?: string;
  editedBy?: string;

  // Timestamps
  createdAt: string;
  supersededAt?: string;
}

//
// MEDIA VAULT - Every file version preserved forever
//

export interface MediaVaultFile {
  id: string;
  chatId: string;
  tenantId: string;
  messageId?: string;
  snapshotId: string;

  // File identity
  originalName: string;                 // What user named it
  displayName: string;                  // What's shown in UI

  // S3 storage with versioning
  s3Bucket: string;
  s3Key: string;
  s3VersionId: string;                  // Critical for immutability

  // File properties
  mimeType: string;
  sizeBytes: number;
  checksumSha256: string;

  // Preview
  thumbnailS3Key?: string;
  previewGenerated: boolean;

  // Version info
  version: number;
```

```
  previousVersionId?: string;

  // Source
  source: 'user_upload' | 'ai_generated' | 'system';

  // Status
  status: MediaStatus;

  // AI-enhanced metadata
  extractedText?: string;              // For searchability
  aiDescription?: string;              // AI-generated description

  // Timestamps
  createdAt: string;
  archivedAt?: string;
}

//
// TIMELINE - Complete history of a chat
//

export interface ChatTimeline {
  chatId: string;
  chatTitle: string;

  // Current state
  currentVersion: number;
  currentMessageCount: number;
  currentFileCount: number;

  // History
  snapshots: TimeMachineSnapshot[];

  // Aggregates
  totalSnapshots: number;
  totalMediaBytes: number;
  oldestSnapshot: string;              // ISO timestamp
  newestSnapshot: string;

  // Calendar data for navigation
  snapshotsByDate: Record<string, number>;  // "2024-12-23" -> count
}

//
// RESTORE REQUEST/RESULT
//
```

```typescript
export interface RestoreRequest {
  chatId: string;
  targetSnapshotId: string;
  scope: RestoreScope;

  // For partial restores
  messageIds?: string[];
  fileIds?: string[];

  // Reason tracking
  reason?: string;
}

export interface RestoreResult {
  success: boolean;
  newSnapshotId: string;

  // What was restored
  messagesRestored: number;
  filesRestored: number;

  // The new current state
  newVersion: number;

  // For undo
  previousSnapshotId: string;
}

//
// EXPORT BUNDLE
//

export interface ExportBundle {
  id: string;
  chatId: string;
  tenantId: string;
  userId: string;

  // Scope
  fromSnapshotId?: string;             // null = from beginning
  toSnapshotId: string;

  // Format
  format: ExportFormat;
  includeMedia: boolean;
```

```
  includeVersionHistory: boolean;

  // File
  s3Key: string;
  sizeBytes: number;
  downloadCount: number;

  // Expiry
  expiresAt: string;

  // Status
  status: 'pending' | 'processing' | 'ready' | 'expired' | 'failed';
  errorMessage?: string;

  // Timestamps
  createdAt: string;
  completedAt?: string;
}
```

32.3 Database Schema

```
-- Migration 013: Time Machine for Think Tank
--

--
-- CHAT SNAPSHOTS - Point-in-time state captures (like Time Machine backups)
--

CREATE TABLE tm_snapshots (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  chat_id UUID NOT NULL REFERENCES thinktank_chats(id) ON DELETE CASCADE,

  -- Version info
  version INTEGER NOT NULL,
  snapshot_timestamp TIMESTAMPTZ NOT NULL DEFAULT NOW(),

  -- State summary
  message_count INTEGER NOT NULL DEFAULT 0,
  file_count INTEGER NOT NULL DEFAULT 0,
  total_tokens BIGINT NOT NULL DEFAULT 0,

  -- Trigger
  trigger TEXT NOT NULL CHECK (trigger IN (
```

```
    'message_sent', 'message_received', 'message_edited', 'message_deleted',
    'file_uploaded', 'file_generated', 'file_deleted', 'chat_renamed',
    'restore_performed', 'manual_snapshot'
  )),
  trigger_message_id UUID,
  trigger_file_id UUID,
  trigger_description TEXT,

  -- Lineage
  previous_snapshot_id UUID REFERENCES tm_snapshots(id),
  restored_from_snapshot_id UUID REFERENCES tm_snapshots(id),

  -- Integrity
  checksum TEXT NOT NULL,

  -- Timestamps
  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

  -- Constraints
  UNIQUE(chat_id, version)
);

-- Indexes for Time Machine navigation
CREATE INDEX idx_tm_snapshots_chat_version ON tm_snapshots(chat_id, version DESC);
CREATE INDEX idx_tm_snapshots_chat_timestamp ON tm_snapshots(chat_id, snapshot_timestamp DES
CREATE INDEX idx_tm_snapshots_date ON tm_snapshots(DATE(snapshot_timestamp), chat_id);

--
-- MESSAGE VERSIONS - Every edit preserved
--

CREATE TABLE tm_message_versions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  message_id UUID NOT NULL,           -- Stable ID across versions
  snapshot_id UUID NOT NULL REFERENCES tm_snapshots(id) ON DELETE CASCADE,

  -- Content
  content TEXT NOT NULL,
  role TEXT NOT NULL CHECK (role IN ('user', 'assistant', 'system')),
  model_id TEXT,

  -- Metadata
  metadata JSONB DEFAULT '{}',

  -- Version info
```

```
  version INTEGER NOT NULL,
  is_active BOOLEAN NOT NULL DEFAULT TRUE,
  is_soft_deleted BOOLEAN NOT NULL DEFAULT FALSE,

  -- Edit tracking
  edit_reason TEXT,
  edited_by UUID REFERENCES users(id),

  -- Timestamps
  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
  superseded_at TIMESTAMPTZ,
  original_created_at TIMESTAMPTZ NOT NULL,  -- When message was first created

  -- Constraints
  UNIQUE(message_id, version)
);

-- Indexes for message lookup
CREATE INDEX idx_tm_messages_message_id ON tm_message_versions(message_id, version DESC);
CREATE INDEX idx_tm_messages_snapshot ON tm_message_versions(snapshot_id);
CREATE INDEX idx_tm_messages_active ON tm_message_versions(message_id) WHERE is_active = TRU
CREATE INDEX idx_tm_messages_search ON tm_message_versions USING gin(to_tsvector('english',

--
-- MEDIA VAULT - Every file version preserved forever
--

CREATE TABLE tm_media_vault (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  chat_id UUID NOT NULL REFERENCES thinktank_chats(id) ON DELETE CASCADE,
  message_id UUID,                    -- Can be NULL for chat-level files
  snapshot_id UUID NOT NULL REFERENCES tm_snapshots(id) ON DELETE CASCADE,

  -- File identity
  original_name TEXT NOT NULL,
  display_name TEXT NOT NULL,

  -- S3 storage (with versioning enabled on bucket)
  s3_bucket TEXT NOT NULL,
  s3_key TEXT NOT NULL,
  s3_version_id TEXT NOT NULL,        -- S3 object version for immutability

  -- File properties
  mime_type TEXT NOT NULL,
  size_bytes BIGINT NOT NULL,
```

```
  checksum_sha256 TEXT NOT NULL,

  -- Preview
  thumbnail_s3_key TEXT,
  preview_generated BOOLEAN DEFAULT FALSE,

  -- Version info
  version INTEGER NOT NULL,
  previous_version_id UUID REFERENCES tm_media_vault(id),

  -- Source
  source TEXT NOT NULL CHECK (source IN ('user_upload', 'ai_generated', 'system')),

  -- Status
  status TEXT NOT NULL DEFAULT 'active' CHECK (status IN (
    'active', 'processing', 'archived', 'soft_deleted'
  )),

  -- AI-enhanced metadata
  extracted_text TEXT,
  ai_description TEXT,
  metadata JSONB DEFAULT '{}',

  -- Timestamps
  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
  archived_at TIMESTAMPTZ,

  -- Constraints
  UNIQUE(chat_id, original_name, version)
);

-- Indexes for media lookup
CREATE INDEX idx_tm_media_chat ON tm_media_vault(chat_id);
CREATE INDEX idx_tm_media_snapshot ON tm_media_vault(snapshot_id);
CREATE INDEX idx_tm_media_name ON tm_media_vault(chat_id, original_name, version DESC);
CREATE INDEX idx_tm_media_search ON tm_media_vault USING gin(
  to_tsvector('english', COALESCE(extracted_text, '') || ' ' || COALESCE(ai_description, ''))
);

--
-- MESSAGE-MEDIA REFERENCES - Links messages to specific file versions
--

CREATE TABLE tm_message_media_refs (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  message_version_id UUID NOT NULL REFERENCES tm_message_versions(id) ON DELETE CASCADE,
```

```
  media_vault_id UUID NOT NULL REFERENCES tm_media_vault(id) ON DELETE CASCADE,

  -- Display order and type
  display_order INTEGER NOT NULL DEFAULT 0,
  reference_type TEXT NOT NULL CHECK (reference_type IN (
    'attachment',      -- User attached this file
    'inline',          -- Embedded in message content
    'result',          -- AI-generated result
    'reference'        -- Referenced but not attached
  )),

  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

  UNIQUE(message_version_id, media_vault_id)
);

--
-- RESTORE LOG - Audit trail for all restores
--

CREATE TABLE tm_restore_log (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  chat_id UUID NOT NULL REFERENCES thinktank_chats(id) ON DELETE CASCADE,
  user_id UUID NOT NULL REFERENCES users(id),

  -- What was restored
  from_snapshot_id UUID NOT NULL REFERENCES tm_snapshots(id),
  to_snapshot_id UUID NOT NULL REFERENCES tm_snapshots(id),

  -- Scope
  scope TEXT NOT NULL CHECK (scope IN (
    'full_chat', 'single_message', 'single_file', 'message_range', 'files_only'
  )),

  -- Items restored
  message_ids UUID[],
  file_ids UUID[],
  messages_restored INTEGER DEFAULT 0,
  files_restored INTEGER DEFAULT 0,

  -- Reason
  reason TEXT,

  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);
```

```sql
--
-- EXPORT BUNDLES - Track export requests
--

CREATE TABLE tm_export_bundles (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  chat_id UUID NOT NULL REFERENCES thinktank_chats(id) ON DELETE CASCADE,
  user_id UUID NOT NULL REFERENCES users(id),

  -- Scope
  from_snapshot_id UUID REFERENCES tm_snapshots(id),
  to_snapshot_id UUID NOT NULL REFERENCES tm_snapshots(id),

  -- Format
  format TEXT NOT NULL CHECK (format IN ('zip', 'json', 'markdown', 'pdf', 'html')),
  include_media BOOLEAN DEFAULT TRUE,
  include_version_history BOOLEAN DEFAULT FALSE,

  -- File
  s3_key TEXT,
  size_bytes BIGINT DEFAULT 0,
  download_count INTEGER DEFAULT 0,

  -- Status
  status TEXT NOT NULL DEFAULT 'pending' CHECK (status IN (
    'pending', 'processing', 'ready', 'expired', 'failed'
  )),
  error_message TEXT,

  -- Expiry
  expires_at TIMESTAMPTZ NOT NULL DEFAULT (NOW() + INTERVAL '7 days'),

  -- Timestamps
  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
  completed_at TIMESTAMPTZ
);

--
-- ROW LEVEL SECURITY
--

ALTER TABLE tm_snapshots ENABLE ROW LEVEL SECURITY;
ALTER TABLE tm_message_versions ENABLE ROW LEVEL SECURITY;
ALTER TABLE tm_media_vault ENABLE ROW LEVEL SECURITY;
```

```sql
ALTER TABLE tm_message_media_refs ENABLE ROW LEVEL SECURITY;
ALTER TABLE tm_restore_log ENABLE ROW LEVEL SECURITY;
ALTER TABLE tm_export_bundles ENABLE ROW LEVEL SECURITY;

-- Tenant isolation policies
CREATE POLICY tm_snapshots_tenant ON tm_snapshots
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tm_message_versions_tenant ON tm_message_versions
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tm_media_vault_tenant ON tm_media_vault
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tm_message_media_refs_tenant ON tm_message_media_refs
  USING (EXISTS (
    SELECT 1 FROM tm_message_versions mv
    WHERE mv.id = tm_message_media_refs.message_version_id
    AND mv.tenant_id = current_setting('app.current_tenant_id')::UUID
  ));

CREATE POLICY tm_restore_log_tenant ON tm_restore_log
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tm_export_bundles_tenant ON tm_export_bundles
  USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

--
-- VIEWS FOR COMMON QUERIES
--

-- Current state view (what users see in normal mode)
CREATE VIEW tm_current_messages AS
SELECT
  mv.message_id,
  mv.content,
  mv.role,
  mv.model_id,
  mv.metadata,
  mv.version,
  mv.original_created_at,
  mv.created_at as version_created_at,
  s.chat_id,
  s.tenant_id
FROM tm_message_versions mv
JOIN tm_snapshots s ON mv.snapshot_id = s.id
```

```sql
WHERE mv.is_active = TRUE AND mv.is_soft_deleted = FALSE;

-- Files with version count
CREATE VIEW tm_files_with_versions AS
SELECT
  mv.*,
  (SELECT COUNT(*) FROM tm_media_vault
   WHERE chat_id = mv.chat_id AND original_name = mv.original_name) as version_count
FROM tm_media_vault mv
WHERE mv.status = 'active';

-- Calendar view for timeline navigation
CREATE VIEW tm_calendar_view AS
SELECT
  chat_id,
  DATE(snapshot_timestamp) as snapshot_date,
  COUNT(*) as snapshot_count,
  MIN(snapshot_timestamp) as first_snapshot,
  MAX(snapshot_timestamp) as last_snapshot
FROM tm_snapshots
GROUP BY chat_id, DATE(snapshot_timestamp)
ORDER BY snapshot_date DESC;
```

32.4 Time Machine Service (Core Business Logic)

```typescript
// packages/functions/src/services/time-machine.service.ts

import { Pool, PoolClient } from 'pg';
import { S3Client, PutObjectCommand, GetObjectCommand, CopyObjectCommand, HeadObjectCommand
import { getSignedUrl } from '@aws-sdk/s3-request-presigner';
import { createHash } from 'crypto';
import { v4 as uuid } from 'uuid';
import {
  TimeMachineSnapshot,
  MessageVersion,
  MediaVaultFile,
  ChatTimeline,
  RestoreRequest,
  RestoreResult,
  ExportBundle,
  SnapshotTrigger,
  RestoreScope,
  ExportFormat,
} from '@radiant/shared';
```

```typescript
//
// TIME MACHINE SERVICE
//

export class TimeMachineService {
  private pool: Pool;
  private s3: S3Client;
  private bucketName: string;

  constructor(pool: Pool) {
    this.pool = pool;
    this.s3 = new S3Client({});
    this.bucketName = process.env.MEDIA_VAULT_BUCKET!;
  }

  //
  // SNAPSHOT CREATION (Automatic on every action)
  //

  async createSnapshot(params: {
    chatId: string;
    tenantId: string;
    trigger: SnapshotTrigger;
    triggerMessageId?: string;
    triggerFileId?: string;
    triggerDescription?: string;
  }): Promise<TimeMachineSnapshot> {
    const client = await this.pool.connect();

    try {
      await client.query('BEGIN');
      await client.query(`SET app.current_tenant_id = '${params.tenantId}'`);

      // Get previous snapshot
      const prevResult = await client.query(`
        SELECT id, version FROM tm_snapshots
        WHERE chat_id = $1
        ORDER BY version DESC LIMIT 1
      `, [params.chatId]);

      const prevSnapshot = prevResult.rows[0];
      const newVersion = prevSnapshot ? prevSnapshot.version + 1 : 1;

      // Count current state
      const countsResult = await client.query(`
```

```
      SELECT
        (SELECT COUNT(DISTINCT message_id) FROM tm_message_versions mv
         JOIN tm_snapshots s ON mv.snapshot_id = s.id
         WHERE s.chat_id = $1 AND mv.is_active = TRUE AND mv.is_soft_deleted = FALSE) as m
        (SELECT COUNT(*) FROM tm_media_vault
         WHERE chat_id = $1 AND status = 'active') as file_count,
        (SELECT COALESCE(SUM((metadata->>'tokens')::bigint), 0) FROM tm_message_versions m
         JOIN tm_snapshots s ON mv.snapshot_id = s.id
         WHERE s.chat_id = $1 AND mv.is_active = TRUE) as total_tokens
    `, [params.chatId]);

    const counts = countsResult.rows[0];

    // Compute checksum of current state
    const checksum = await this.computeChatChecksum(client, params.chatId);

    // Create snapshot
    const result = await client.query(`
      INSERT INTO tm_snapshots (
        tenant_id, chat_id, version, message_count, file_count, total_tokens,
        trigger, trigger_message_id, trigger_file_id, trigger_description,
        previous_snapshot_id, checksum
      ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12)
      RETURNING *
    `, [
      params.tenantId,
      params.chatId,
      newVersion,
      parseInt(counts.message_count) || 0,
      parseInt(counts.file_count) || 0,
      parseInt(counts.total_tokens) || 0,
      params.trigger,
      params.triggerMessageId,
      params.triggerFileId,
      params.triggerDescription,
      prevSnapshot?.id,
      checksum,
    ]);

    await client.query('COMMIT');

    return this.mapSnapshotRow(result.rows[0]);
  } catch (error) {
    await client.query('ROLLBACK');
    throw error;
  } finally {
```

```
    client.release();
  }
}

private async computeChatChecksum(client: PoolClient, chatId: string): Promise<string> {
  const result = await client.query(`
    SELECT mv.message_id, mv.content, mv.role, mv.version
    FROM tm_message_versions mv
    JOIN tm_snapshots s ON mv.snapshot_id = s.id
    WHERE s.chat_id = $1 AND mv.is_active = TRUE AND mv.is_soft_deleted = FALSE
    ORDER BY mv.original_created_at ASC
  `, [chatId]);

  const hash = createHash('sha256');
  for (const row of result.rows) {
    hash.update(`${row.message_id}:${row.content}:${row.role}:${row.version}|`);
  }
  return hash.digest('hex');
}

//
// MESSAGE VERSIONING
//

async saveMessageVersion(params: {
  chatId: string;
  tenantId: string;
  messageId: string;
  content: string;
  role: 'user' | 'assistant' | 'system';
  modelId?: string;
  metadata?: Record<string, unknown>;
  isEdit?: boolean;
  editReason?: string;
  editedBy?: string;
}): Promise<MessageVersion> {
  const client = await this.pool.connect();

  try {
    await client.query('BEGIN');
    await client.query(`SET app.current_tenant_id = '${params.tenantId}'`);

    // Get or create snapshot
    let snapshot = await this.getLatestSnapshot(client, params.chatId);
    if (!snapshot) {
      // Create initial snapshot
```

```
  await client.query('COMMIT');
  snapshot = await this.createSnapshot({
    chatId: params.chatId,
    tenantId: params.tenantId,
    trigger: params.role === 'user' ? 'message_sent' : 'message_received',
  });
  await client.query('BEGIN');
  await client.query(`SET app.current_tenant_id = '${params.tenantId}'`);
}

// Get previous version of this message (if editing)
const prevResult = await client.query(`
  SELECT id, version FROM tm_message_versions
  WHERE message_id = $1 AND is_active = TRUE
  ORDER BY version DESC LIMIT 1
`, [params.messageId]);

const prevVersion = prevResult.rows[0];
const newVersion = prevVersion ? prevVersion.version + 1 : 1;

// If editing, mark previous as superseded
if (prevVersion) {
  await client.query(`
    UPDATE tm_message_versions
    SET is_active = FALSE, superseded_at = NOW()
    WHERE id = $1
  `, [prevVersion.id]);
}

// Insert new version
const result = await client.query(`
  INSERT INTO tm_message_versions (
    tenant_id, message_id, snapshot_id, content, role, model_id,
    metadata, version, is_active, edit_reason, edited_by, original_created_at
  ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, TRUE, $9, $10,
    COALESCE((SELECT original_created_at FROM tm_message_versions WHERE message_id = $
  )
  RETURNING *
`, [
  params.tenantId,
  params.messageId,
  snapshot.id,
  params.content,
  params.role,
  params.modelId,
  JSON.stringify(params.metadata || {}),
```

```
        newVersion,
        params.editReason,
        params.editedBy,
      ]);

      await client.query('COMMIT');

      // Create snapshot for this change
      await this.createSnapshot({
        chatId: params.chatId,
        tenantId: params.tenantId,
        trigger: params.isEdit ? 'message_edited' : (params.role === 'user' ? 'message_sent'
        triggerMessageId: params.messageId,
      });

      return this.mapMessageVersionRow(result.rows[0]);
    } catch (error) {
      await client.query('ROLLBACK');
      throw error;
    } finally {
      client.release();
    }
  }

  async softDeleteMessage(params: {
    chatId: string;
    tenantId: string;
    messageId: string;
    deletedBy: string;
  }): Promise<void> {
    const client = await this.pool.connect();

    try {
      await client.query('BEGIN');
      await client.query(`SET app.current_tenant_id = '${params.tenantId}'`);

      // Mark as soft deleted (NEVER actually delete)
      await client.query(`
        UPDATE tm_message_versions
        SET is_soft_deleted = TRUE, superseded_at = NOW()
        WHERE message_id = $1 AND is_active = TRUE
      `, [params.messageId]);

      await client.query('COMMIT');

      // Create snapshot
```

```
    await this.createSnapshot({
      chatId: params.chatId,
      tenantId: params.tenantId,
      trigger: 'message_deleted',
      triggerMessageId: params.messageId,
    });
  } catch (error) {
    await client.query('ROLLBACK');
    throw error;
  } finally {
    client.release();
  }
}

//
// MEDIA VAULT
//

async uploadFile(params: {
  chatId: string;
  tenantId: string;
  messageId?: string;
  file: {
    name: string;
    data: Buffer;
    mimeType: string;
  };
  source: 'user_upload' | 'ai_generated' | 'system';
}): Promise<MediaVaultFile> {
  const client = await this.pool.connect();

  try {
    await client.query('BEGIN');
    await client.query(`SET app.current_tenant_id = '${params.tenantId}'`);

    // Get or create snapshot
    let snapshot = await this.getLatestSnapshot(client, params.chatId);
    if (!snapshot) {
      await client.query('COMMIT');
      snapshot = await this.createSnapshot({
        chatId: params.chatId,
        tenantId: params.tenantId,
        trigger: 'file_uploaded',
      });
      await client.query('BEGIN');
      await client.query(`SET app.current_tenant_id = '${params.tenantId}'`);
```

```
}

// Check for existing versions
const existingResult = await client.query(`
  SELECT id, version FROM tm_media_vault
  WHERE chat_id = $1 AND original_name = $2
  ORDER BY version DESC LIMIT 1
`, [params.chatId, params.file.name]);

const existing = existingResult.rows[0];
const newVersion = existing ? existing.version + 1 : 1;

// Compute checksum
const checksum = createHash('sha256').update(params.file.data).digest('hex');

// Generate S3 key
const fileId = uuid();
const s3Key = `${params.tenantId}/${params.chatId}/${fileId}/${params.file.name}`;

// Upload to S3 (bucket has versioning enabled)
const putResult = await this.s3.send(new PutObjectCommand({
  Bucket: this.bucketName,
  Key: s3Key,
  Body: params.file.data,
  ContentType: params.file.mimeType,
  Metadata: {
    'chat-id': params.chatId,
    'original-name': params.file.name,
    'version': String(newVersion),
    'checksum': checksum,
  },
}));

// Insert into media vault
const result = await client.query(`
  INSERT INTO tm_media_vault (
    tenant_id, chat_id, message_id, snapshot_id, original_name, display_name,
    s3_bucket, s3_key, s3_version_id, mime_type, size_bytes, checksum_sha256,
    version, previous_version_id, source
  ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15)
  RETURNING *
`, [
  params.tenantId,
  params.chatId,
  params.messageId,
  snapshot.id,
```

```
        params.file.name,
        params.file.name,
        this.bucketName,
        s3Key,
        putResult.VersionId!,
        params.file.mimeType,
        params.file.data.length,
        checksum,
        newVersion,
        existing?.id,
        params.source,
      ]);

      await client.query('COMMIT');

      // Create snapshot
      await this.createSnapshot({
        chatId: params.chatId,
        tenantId: params.tenantId,
        trigger: params.source === 'user_upload' ? 'file_uploaded' : 'file_generated',
        triggerFileId: result.rows[0].id,
      });

      return this.mapMediaVaultRow(result.rows[0]);
    } catch (error) {
      await client.query('ROLLBACK');
      throw error;
    } finally {
      client.release();
    }
  }

  async getFileDownloadUrl(fileId: string, expiresIn = 3600): Promise<string> {
    const result = await this.pool.query(`
      SELECT s3_bucket, s3_key, s3_version_id FROM tm_media_vault WHERE id = $1
    `, [fileId]);

    if (!result.rows[0]) {
      throw new Error('File not found');
    }

    const { s3_bucket, s3_key, s3_version_id } = result.rows[0];

    const command = new GetObjectCommand({
      Bucket: s3_bucket,
      Key: s3_key,
```

```
    VersionId: s3_version_id,
  });

  return getSignedUrl(this.s3, command, { expiresIn });
}

async getFileVersions(chatId: string, fileName: string): Promise<MediaVaultFile[]> {
  const result = await this.pool.query(`
    SELECT * FROM tm_media_vault
    WHERE chat_id = $1 AND original_name = $2
    ORDER BY version DESC
  `, [chatId, fileName]);

  return result.rows.map(row => this.mapMediaVaultRow(row));
}

//
// TIMELINE NAVIGATION (The "fly back through time" experience)
//

async getTimeline(chatId: string, tenantId: string): Promise<ChatTimeline> {
  await this.pool.query(`SET app.current_tenant_id = '${tenantId}'`);

  // Get chat info
  const chatResult = await this.pool.query(`
    SELECT title FROM thinktank_chats WHERE id = $1
  `, [chatId]);

  // Get all snapshots
  const snapshotsResult = await this.pool.query(`
    SELECT * FROM tm_snapshots
    WHERE chat_id = $1
    ORDER BY version ASC
  `, [chatId]);

  // Get calendar data
  const calendarResult = await this.pool.query(`
    SELECT snapshot_date::text, snapshot_count
    FROM tm_calendar_view
    WHERE chat_id = $1
  `, [chatId]);

  // Get total media size
  const sizeResult = await this.pool.query(`
    SELECT COALESCE(SUM(size_bytes), 0) as total_size
    FROM tm_media_vault
```

```
      WHERE chat_id = $1
    `, [chatId]);

    const snapshots = snapshotsResult.rows.map(row => this.mapSnapshotRow(row));
    const currentSnapshot = snapshots[snapshots.length - 1];

    const snapshotsByDate: Record<string, number> = {};
    for (const row of calendarResult.rows) {
      snapshotsByDate[row.snapshot_date] = parseInt(row.snapshot_count);
    }

    return {
      chatId,
      chatTitle: chatResult.rows[0]?.title || 'Untitled Chat',
      currentVersion: currentSnapshot?.version || 0,
      currentMessageCount: currentSnapshot?.messageCount || 0,
      currentFileCount: currentSnapshot?.fileCount || 0,
      snapshots,
      totalSnapshots: snapshots.length,
      totalMediaBytes: parseInt(sizeResult.rows[0].total_size) || 0,
      oldestSnapshot: snapshots[0]?.timestamp || new Date().toISOString(),
      newestSnapshot: currentSnapshot?.timestamp || new Date().toISOString(),
      snapshotsByDate,
    };
  }

  async getChatAtSnapshot(chatId: string, snapshotId: string, tenantId: string): Promise<{
    snapshot: TimeMachineSnapshot;
    messages: MessageVersion[];
    files: MediaVaultFile[];
  }> {
    await this.pool.query(`SET app.current_tenant_id = '${tenantId}'`);

    // Get snapshot
    const snapshotResult = await this.pool.query(`
      SELECT * FROM tm_snapshots WHERE id = $1
    `, [snapshotId]);

    if (!snapshotResult.rows[0]) {
      throw new Error('Snapshot not found');
    }

    // Get messages at this snapshot
    // This requires understanding the chain - we need messages that were active AT this sna
    const messagesResult = await this.pool.query(`
      WITH snapshot_chain AS (
```

```
      SELECT id, version FROM tm_snapshots
      WHERE chat_id = $1 AND version <= (SELECT version FROM tm_snapshots WHERE id = $2)
    )
    SELECT DISTINCT ON (mv.message_id) mv.*
    FROM tm_message_versions mv
    WHERE mv.snapshot_id IN (SELECT id FROM snapshot_chain)
      AND NOT mv.is_soft_deleted
    ORDER BY mv.message_id, mv.version DESC
  `, [chatId, snapshotId]);

  // Get files at this snapshot
  const filesResult = await this.pool.query(`
    WITH snapshot_chain AS (
      SELECT id, version FROM tm_snapshots
      WHERE chat_id = $1 AND version <= (SELECT version FROM tm_snapshots WHERE id = $2)
    )
    SELECT DISTINCT ON (mf.original_name) mf.*
    FROM tm_media_vault mf
    WHERE mf.snapshot_id IN (SELECT id FROM snapshot_chain)
      AND mf.status != 'soft_deleted'
    ORDER BY mf.original_name, mf.version DESC
  `, [chatId, snapshotId]);

  return {
    snapshot: this.mapSnapshotRow(snapshotResult.rows[0]),
    messages: messagesResult.rows.map(row => this.mapMessageVersionRow(row)),
    files: filesResult.rows.map(row => this.mapMediaVaultRow(row)),
  };
}

async getSnapshotsByDate(chatId: string, date: string, tenantId: string): Promise<TimeMach
  await this.pool.query(`SET app.current_tenant_id = '${tenantId}'`);

  const result = await this.pool.query(`
    SELECT * FROM tm_snapshots
    WHERE chat_id = $1 AND DATE(snapshot_timestamp) = $2
    ORDER BY snapshot_timestamp ASC
  `, [chatId, date]);

  return result.rows.map(row => this.mapSnapshotRow(row));
}

//
// RESTORE (One-click recovery)
//
```

```
async restore(request: RestoreRequest, userId: string, tenantId: string): Promise<RestoreR
  const client = await this.pool.connect();

  try {
    await client.query('BEGIN');
    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    // Get target snapshot state
    const targetState = await this.getChatAtSnapshot(request.chatId, request.targetSnapsho

    // Get current snapshot for logging
    const currentSnapshot = await this.getLatestSnapshot(client, request.chatId);

    let messagesRestored = 0;
    let filesRestored = 0;

    switch (request.scope) {
      case 'full_chat':
        // Restore all messages and files
        messagesRestored = await this.restoreMessages(client, targetState.messages, reques
        filesRestored = await this.restoreFiles(client, targetState.files, request.chatId,
        break;

      case 'single_message':
        if (request.messageIds?.length) {
          const targetMessages = targetState.messages.filter(m => request.messageIds!.incl
          messagesRestored = await this.restoreMessages(client, targetMessages, request.ch
        }
        break;

      case 'single_file':
        if (request.fileIds?.length) {
          const targetFiles = targetState.files.filter(f => request.fileIds!.includes(f.id
          filesRestored = await this.restoreFiles(client, targetFiles, request.chatId, ten
        }
        break;

      case 'files_only':
        filesRestored = await this.restoreFiles(client, targetState.files, request.chatId,
        break;
    }

    await client.query('COMMIT');

    // Create restore snapshot
    const newSnapshot = await this.createSnapshot({
```

```
        chatId: request.chatId,
        tenantId,
        trigger: 'restore_performed',
        triggerDescription: `Restored to version ${targetState.snapshot.version}`,
      });

      // Log the restore
      await this.pool.query(`
        INSERT INTO tm_restore_log (
          tenant_id, chat_id, user_id, from_snapshot_id, to_snapshot_id,
          scope, message_ids, file_ids, messages_restored, files_restored, reason
        ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11)
      `, [
        tenantId,
        request.chatId,
        userId,
        request.targetSnapshotId,
        newSnapshot.id,
        request.scope,
        request.messageIds || [],
        request.fileIds || [],
        messagesRestored,
        filesRestored,
        request.reason,
      ]);

      return {
        success: true,
        newSnapshotId: newSnapshot.id,
        messagesRestored,
        filesRestored,
        newVersion: newSnapshot.version,
        previousSnapshotId: currentSnapshot?.id || '',
      };
    } catch (error) {
      await client.query('ROLLBACK');
      throw error;
    } finally {
      client.release();
    }
  }

  private async restoreMessages(
    client: PoolClient,
    messages: MessageVersion[],
    chatId: string,
```

```
  tenantId: string
): Promise<number> {
  // Deactivate current versions
  await client.query(`
    UPDATE tm_message_versions
    SET is_active = FALSE, superseded_at = NOW()
    WHERE message_id IN (
      SELECT DISTINCT message_id FROM tm_message_versions mv
      JOIN tm_snapshots s ON mv.snapshot_id = s.id
      WHERE s.chat_id = $1 AND mv.is_active = TRUE
    )
  `, [chatId]);

  // Get latest snapshot
  const snapshot = await this.getLatestSnapshot(client, chatId);

  // Insert restored versions as new active versions
  for (const msg of messages) {
    await client.query(`
      INSERT INTO tm_message_versions (
        tenant_id, message_id, snapshot_id, content, role, model_id,
        metadata, version, is_active, edit_reason, original_created_at
      ) VALUES ($1, $2, $3, $4, $5, $6, $7,
        (SELECT COALESCE(MAX(version), 0) + 1 FROM tm_message_versions WHERE message_id =
        TRUE, 'Restored from Time Machine', $8)
    `, [
      tenantId,
      msg.messageId,
      snapshot?.id,
      msg.content,
      msg.role,
      msg.modelId,
      JSON.stringify(msg.metadata || {}),
      msg.createdAt,
    ]);
  }

  return messages.length;
}

private async restoreFiles(
  client: PoolClient,
  files: MediaVaultFile[],
  chatId: string,
  tenantId: string
): Promise<number> {
```

```
  // Mark current files as soft deleted
  await client.query(`
    UPDATE tm_media_vault
    SET status = 'soft_deleted'
    WHERE chat_id = $1 AND status = 'active'
  `, [chatId]);

  // Get latest snapshot
  const snapshot = await this.getLatestSnapshot(client, chatId);

  // "Restore" files by creating new active versions pointing to same S3 objects
  for (const file of files) {
    await client.query(`
      INSERT INTO tm_media_vault (
        tenant_id, chat_id, message_id, snapshot_id, original_name, display_name,
        s3_bucket, s3_key, s3_version_id, mime_type, size_bytes, checksum_sha256,
        version, previous_version_id, source, status, extracted_text, ai_description
      ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12,
        (SELECT COALESCE(MAX(version), 0) + 1 FROM tm_media_vault WHERE chat_id = $2 AND o
        $13, $14, 'active', $15, $16)
    `, [
      tenantId,
      chatId,
      file.messageId,
      snapshot?.id,
      file.originalName,
      file.displayName,
      file.s3Bucket,
      file.s3Key,
      file.s3VersionId,
      file.mimeType,
      file.sizeBytes,
      file.checksumSha256,
      file.id,
      file.source,
      file.extractedText,
      file.aiDescription,
    ]);
  }

  return files.length;
}

//
// SEARCH (Find that thing from 3 months ago)
//
```

```
async searchMessages(chatId: string, query: string, tenantId: string): Promise<MessageVers
  await this.pool.query(`SET app.current_tenant_id = '${tenantId}'`);

  const result = await this.pool.query(`
    SELECT DISTINCT ON (mv.message_id) mv.*,
      ts_rank(to_tsvector('english', mv.content), plainto_tsquery('english', $2)) as rank
    FROM tm_message_versions mv
    JOIN tm_snapshots s ON mv.snapshot_id = s.id
    WHERE s.chat_id = $1
      AND to_tsvector('english', mv.content) @@ plainto_tsquery('english', $2)
    ORDER BY mv.message_id, rank DESC, mv.version DESC
    LIMIT 50
  `, [chatId, query]);

  return result.rows.map(row => this.mapMessageVersionRow(row));
}

async searchFiles(chatId: string, query: string, tenantId: string): Promise<MediaVaultFile
  await this.pool.query(`SET app.current_tenant_id = '${tenantId}'`);

  const result = await this.pool.query(`
    SELECT DISTINCT ON (original_name) *,
      ts_rank(
        to_tsvector('english', COALESCE(extracted_text, '') || ' ' || COALESCE(ai_descript
        plainto_tsquery('english', $2)
      ) as rank
    FROM tm_media_vault
    WHERE chat_id = $1
      AND (
        original_name ILIKE '%' || $2 || '%'
        OR to_tsvector('english', COALESCE(extracted_text, '') || ' ' || COALESCE(ai_descr
           @@ plainto_tsquery('english', $2)
      )
    ORDER BY original_name, rank DESC, version DESC
    LIMIT 50
  `, [chatId, query]);

  return result.rows.map(row => this.mapMediaVaultRow(row));
}

//
// EXPORT
//

async createExportBundle(params: {
```

```
  chatId: string;
  tenantId: string;
  userId: string;
  format: ExportFormat;
  includeMedia: boolean;
  includeVersionHistory: boolean;
  fromSnapshotId?: string;
}): Promise<string> {
  // Get current snapshot
  const currentResult = await this.pool.query(`
    SELECT id FROM tm_snapshots
    WHERE chat_id = $1
    ORDER BY version DESC LIMIT 1
  `, [params.chatId]);

  const bundleId = uuid();

  await this.pool.query(`
    INSERT INTO tm_export_bundles (
      id, tenant_id, chat_id, user_id, from_snapshot_id, to_snapshot_id,
      format, include_media, include_version_history, status
    ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, 'pending')
  `, [
    bundleId,
    params.tenantId,
    params.chatId,
    params.userId,
    params.fromSnapshotId,
    currentResult.rows[0]?.id,
    params.format,
    params.includeMedia,
    params.includeVersionHistory,
  ]);

  // Trigger async export via SQS - see Section 33.5 for export queue handler
  // await sqs.send(new SendMessageCommand({
  //   QueueUrl: process.env.EXPORT_QUEUE_URL,
  //   MessageBody: JSON.stringify({ chatId, format, includeMedia }),
  // }));

  return bundleId;
}

//
// HELPERS
//
```

```
private async getLatestSnapshot(client: PoolClient, chatId: string): Promise<TimeMachineSn
  const result = await client.query(`
    SELECT * FROM tm_snapshots
    WHERE chat_id = $1
    ORDER BY version DESC LIMIT 1
  `, [chatId]);

  return result.rows[0] ? this.mapSnapshotRow(result.rows[0]) : null;
}

private mapSnapshotRow(row: any): TimeMachineSnapshot {
  return {
    id: row.id,
    chatId: row.chat_id,
    tenantId: row.tenant_id,
    version: row.version,
    timestamp: row.snapshot_timestamp,
    messageCount: row.message_count,
    fileCount: row.file_count,
    totalTokens: row.total_tokens,
    trigger: row.trigger,
    triggerDetails: {
      messageId: row.trigger_message_id,
      fileId: row.trigger_file_id,
      description: row.trigger_description,
    },
    previousSnapshotId: row.previous_snapshot_id,
    restoredFromSnapshotId: row.restored_from_snapshot_id,
    checksum: row.checksum,
    createdAt: row.created_at,
  };
}

private mapMessageVersionRow(row: any): MessageVersion {
  return {
    id: row.id,
    messageId: row.message_id,
    tenantId: row.tenant_id,
    snapshotId: row.snapshot_id,
    content: row.content,
    role: row.role,
    modelId: row.model_id,
    version: row.version,
    isActive: row.is_active,
    isSoftDeleted: row.is_soft_deleted,
```

```
      editReason: row.edit_reason,
      editedBy: row.edited_by,
      createdAt: row.created_at,
      supersededAt: row.superseded_at,
    };
  }

  private mapMediaVaultRow(row: any): MediaVaultFile {
    return {
      id: row.id,
      chatId: row.chat_id,
      tenantId: row.tenant_id,
      messageId: row.message_id,
      snapshotId: row.snapshot_id,
      originalName: row.original_name,
      displayName: row.display_name,
      s3Bucket: row.s3_bucket,
      s3Key: row.s3_key,
      s3VersionId: row.s3_version_id,
      mimeType: row.mime_type,
      sizeBytes: row.size_bytes,
      checksumSha256: row.checksum_sha256,
      thumbnailS3Key: row.thumbnail_s3_key,
      previewGenerated: row.preview_generated,
      version: row.version,
      previousVersionId: row.previous_version_id,
      source: row.source,
      status: row.status,
      extractedText: row.extracted_text,
      aiDescription: row.ai_description,
      createdAt: row.created_at,
      archivedAt: row.archived_at,
    };
  }
}
```

32.5 Complex API Handlers (Service Layer Exposure)

```
// packages/functions/src/handlers/thinktank/time-machine.handlers.ts

import { APIGatewayProxyEvent, APIGatewayProxyResult } from 'aws-lambda';
import { TimeMachineService } from '../../services/time-machine.service';
import { pool } from '../../utils/db';
import { RestoreScope, ExportFormat } from '@radiant/shared';
```

```
const service = new TimeMachineService(pool);

const corsHeaders = {
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Headers': 'Content-Type,Authorization',
  'Content-Type': 'application/json',
};

function getUserContext(event: APIGatewayProxyEvent) {
  return {
    userId: event.requestContext.authorizer?.claims?.sub,
    tenantId: event.requestContext.authorizer?.claims?.['custom:tenant_id'],
  };
}

//
// TIMELINE ENDPOINTS
//

// GET /api/thinktank/chats/:chatId/time-machine
export async function getTimeline(event: APIGatewayProxyEvent): Promise<APIGatewayProxyResul
  try {
    const { tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;

    if (!chatId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const timeline = await service.getTimeline(chatId, tenantId);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify(timeline),
    };
  } catch (error: any) {
    console.error('getTimeline error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

// GET /api/thinktank/chats/:chatId/time-machine/snapshots/:snapshotId
export async function getChatAtSnapshot(event: APIGatewayProxyEvent): Promise<APIGatewayProx
  try {
```

```
    const { tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;
    const snapshotId = event.pathParameters?.snapshotId;

    if (!chatId || !snapshotId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const state = await service.getChatAtSnapshot(chatId, snapshotId, tenantId);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify(state),
    };
  } catch (error: any) {
    console.error('getChatAtSnapshot error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

// GET /api/thinktank/chats/:chatId/time-machine/calendar/:date
export async function getSnapshotsByDate(event: APIGatewayProxyEvent): Promise<APIGatewayPro
  try {
    const { tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;
    const date = event.pathParameters?.date;  // YYYY-MM-DD

    if (!chatId || !date) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const snapshots = await service.getSnapshotsByDate(chatId, date, tenantId);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({ snapshots }),
    };
  } catch (error: any) {
    console.error('getSnapshotsByDate error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

//
```

```
// RESTORE ENDPOINTS
//

// POST /api/thinktank/chats/:chatId/time-machine/restore
export async function restoreFromSnapshot(event: APIGatewayProxyEvent): Promise<APIGatewayPr
  try {
    const { userId, tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;
    const body = JSON.parse(event.body || '{}');

    if (!chatId || !body.snapshotId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const result = await service.restore({
      chatId,
      targetSnapshotId: body.snapshotId,
      scope: (body.scope || 'full_chat') as RestoreScope,
      messageIds: body.messageIds,
      fileIds: body.fileIds,
      reason: body.reason,
    }, userId, tenantId);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify(result),
    };
  } catch (error: any) {
    console.error('restoreFromSnapshot error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

//
// MEDIA VAULT ENDPOINTS
//

// GET /api/thinktank/chats/:chatId/time-machine/files
export async function getFiles(event: APIGatewayProxyEvent): Promise<APIGatewayProxyResult>
  try {
    const { tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;

    if (!chatId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
```

```
    }

    const result = await pool.query(`
      SELECT * FROM tm_files_with_versions
      WHERE chat_id = $1
      ORDER BY created_at DESC
    `, [chatId]);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({ files: result.rows }),
    };
  } catch (error: any) {
    console.error('getFiles error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

// GET /api/thinktank/chats/:chatId/time-machine/files/:fileName/versions
export async function getFileVersions(event: APIGatewayProxyEvent): Promise<APIGatewayProxyR
  try {
    const { tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;
    const fileName = decodeURIComponent(event.pathParameters?.fileName || '');

    if (!chatId || !fileName) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const versions = await service.getFileVersions(chatId, fileName);

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({ versions }),
    };
  } catch (error: any) {
    console.error('getFileVersions error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

// GET /api/thinktank/files/:fileId/download
export async function downloadFile(event: APIGatewayProxyEvent): Promise<APIGatewayProxyResu
  try {
```

```
    const fileId = event.pathParameters?.fileId;

    if (!fileId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'fileId
    }

    const url = await service.getFileDownloadUrl(fileId);

    return {
      statusCode: 302,
      headers: { ...corsHeaders, Location: url },
      body: '',
    };
  } catch (error: any) {
    console.error('downloadFile error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

//
// SEARCH ENDPOINTS
//

// GET /api/thinktank/chats/:chatId/time-machine/search
export async function search(event: APIGatewayProxyEvent): Promise<APIGatewayProxyResult> {
  try {
    const { tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;
    const query = event.queryStringParameters?.q;
    const type = event.queryStringParameters?.type || 'all';  // 'messages', 'files', 'all'

    if (!chatId || !query) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const results: { messages?: any[]; files?: any[] } = {};

    if (type === 'all' || type === 'messages') {
      results.messages = await service.searchMessages(chatId, query, tenantId);
    }

    if (type === 'all' || type === 'files') {
      results.files = await service.searchFiles(chatId, query, tenantId);
    }

    return {
```

```
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify(results),
    };
  } catch (error: any) {
    console.error('search error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

//
// EXPORT ENDPOINTS
//

// POST /api/thinktank/chats/:chatId/time-machine/export
export async function createExport(event: APIGatewayProxyEvent): Promise<APIGatewayProxyResu
  try {
    const { userId, tenantId } = getUserContext(event);
    const chatId = event.pathParameters?.chatId;
    const body = JSON.parse(event.body || '{}');

    if (!chatId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const bundleId = await service.createExportBundle({
      chatId,
      tenantId,
      userId,
      format: (body.format || 'zip') as ExportFormat,
      includeMedia: body.includeMedia !== false,
      includeVersionHistory: body.includeVersionHistory === true,
      fromSnapshotId: body.fromSnapshotId,
    });

    return {
      statusCode: 202,
      headers: corsHeaders,
      body: JSON.stringify({
        bundleId,
        status: 'pending',
        message: 'Export is being prepared. Check status at /api/thinktank/exports/:bundleId
      }),
    };
  } catch (error: any) {
    console.error('createExport error:', error);
```

```
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

// GET /api/thinktank/exports/:bundleId
export async function getExportStatus(event: APIGatewayProxyEvent): Promise<APIGatewayProxyR
  try {
    const bundleId = event.pathParameters?.bundleId;

    if (!bundleId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'bundleI
    }

    const result = await pool.query(`
      SELECT * FROM tm_export_bundles WHERE id = $1
    `, [bundleId]);

    if (!result.rows[0]) {
      return { statusCode: 404, headers: corsHeaders, body: JSON.stringify({ error: 'Export
    }

    const bundle = result.rows[0];

    // If ready, generate download URL
    let downloadUrl: string | undefined;
    if (bundle.status === 'ready' && bundle.s3_key) {
      downloadUrl = await service.getFileDownloadUrl(bundle.s3_key);
    }

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({
        bundleId: bundle.id,
        status: bundle.status,
        format: bundle.format,
        sizeBytes: bundle.size_bytes,
        downloadUrl,
        expiresAt: bundle.expires_at,
        createdAt: bundle.created_at,
        completedAt: bundle.completed_at,
        errorMessage: bundle.error_message,
      }),
    };
  } catch (error: any) {
    console.error('getExportStatus error:', error);
```

```
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}
```

32.6 API Routes Configuration

```
// packages/functions/src/routes/time-machine.routes.ts

import { Router } from './router';
import * as handlers from '../handlers/thinktank/time-machine.handlers';

export function registerTimeMachineRoutes(router: Router) {
  // Timeline
  router.get('/api/thinktank/chats/:chatId/time-machine', handlers.getTimeline);
  router.get('/api/thinktank/chats/:chatId/time-machine/snapshots/:snapshotId', handlers.get
  router.get('/api/thinktank/chats/:chatId/time-machine/calendar/:date', handlers.getSnapsho

  // Restore
  router.post('/api/thinktank/chats/:chatId/time-machine/restore', handlers.restoreFromSnaps

  // Files
  router.get('/api/thinktank/chats/:chatId/time-machine/files', handlers.getFiles);
  router.get('/api/thinktank/chats/:chatId/time-machine/files/:fileName/versions', handlers.
  router.get('/api/thinktank/files/:fileId/download', handlers.downloadFile);

  // Search
  router.get('/api/thinktank/chats/:chatId/time-machine/search', handlers.search);

  // Export
  router.post('/api/thinktank/chats/:chatId/time-machine/export', handlers.createExport);
  router.get('/api/thinktank/exports/:bundleId', handlers.getExportStatus);
}
```

SECTION-33-TIME-MACHINE-UI

SECTION 33: TIME MACHINE UI & SIMPLIFIED AI API (v4.0.0)

Version: 4.0.0 | The visual "fly back through time" experience + AI-friendly API

33.1 Simplified AI API for Time Machine

The AI API allows client apps to let their AI assistants help users navigate history naturally.

```
// packages/functions/src/handlers/thinktank/ai-time-machine.handlers.ts

import { APIGatewayProxyEvent, APIGatewayProxyResult } from 'aws-lambda';
import { TimeMachineService } from '../../services/time-machine.service';
import { pool } from '../../utils/db';

const service = new TimeMachineService(pool);

const corsHeaders = {
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Headers': 'Content-Type,Authorization',
  'Content-Type': 'application/json',
};

//
// AI-FRIENDLY SIMPLIFIED API
// These endpoints return human-readable summaries that AI can relay to users
//

/**
 * GET /api/ai/chats/:chatId/history/summary
 *
 * Returns a human-readable summary of chat history that an AI can present.
 *
 * Example response:
 * {
 *   "summary": "This conversation has 47 snapshots over 3 days. You've exchanged
 *               156 messages and shared 8 files. The oldest point you can restore
 *               to is December 20th at 2:34 PM.",
```

```
 *   "highlights": [
 *     "December 22: Major file update - report_final.xlsx",
 *     "December 21: Long discussion about project requirements",
 *     "December 20: Conversation started"
 *   ],
 *   "canRestore": true,
 *   "oldestDate": "2024-12-20",
 *   "newestDate": "2024-12-23"
 * }
 */
export async function getHistorySummary(event: APIGatewayProxyEvent): Promise<APIGatewayProx
  try {
    const chatId = event.pathParameters?.chatId;
    const tenantId = event.requestContext.authorizer?.claims?.['custom:tenant_id'];

    if (!chatId) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    const timeline = await service.getTimeline(chatId, tenantId);

    // Generate human-readable summary
    const dayCount = Object.keys(timeline.snapshotsByDate).length;
    const oldestDate = new Date(timeline.oldestSnapshot).toLocaleDateString('en-US', {
      month: 'long', day: 'numeric', hour: 'numeric', minute: '2-digit'
    });

    // Generate highlights from significant snapshots
    const highlights: string[] = [];
    const fileSnapshots = timeline.snapshots.filter(s =>
      s.trigger === 'file_uploaded' || s.trigger === 'file_generated'
    );
    const restoreSnapshots = timeline.snapshots.filter(s => s.trigger === 'restore_performed

    // Add recent file activity
    if (fileSnapshots.length > 0) {
      const recent = fileSnapshots[fileSnapshots.length - 1];
      const date = new Date(recent.timestamp).toLocaleDateString('en-US', { month: 'long', d
      highlights.push(`${date}: File activity (${timeline.currentFileCount} files total)`);
    }

    // Add restore activity
    if (restoreSnapshots.length > 0) {
      highlights.push(`You've restored from history ${restoreSnapshots.length} time(s)`);
    }
```

```
    // Add conversation start
    if (timeline.snapshots.length > 0) {
      const first = timeline.snapshots[0];
      const date = new Date(first.timestamp).toLocaleDateString('en-US', { month: 'long', da
      highlights.push(`${date}: Conversation started`);
    }

    const summary = `This conversation has ${timeline.totalSnapshots} snapshots over ${dayCo
      `You've exchanged ${timeline.currentMessageCount} messages and shared ${timeline.curre
      `The oldest point you can restore to is ${oldestDate}.`;

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({
        summary,
        highlights,
        canRestore: timeline.totalSnapshots > 1,
        oldestDate: timeline.oldestSnapshot.split('T')[0],
        newestDate: timeline.newestSnapshot.split('T')[0],
        stats: {
          totalSnapshots: timeline.totalSnapshots,
          messageCount: timeline.currentMessageCount,
          fileCount: timeline.currentFileCount,
          totalMediaBytes: timeline.totalMediaBytes,
        },
      }),
    };
  } catch (error: any) {
    console.error('getHistorySummary error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

/**
 * POST /api/ai/chats/:chatId/history/find
 *
 * Natural language search through history.
 *
 * Request:
 * { "query": "that spreadsheet from last week" }
 *
 * Response:
 * {
 *   "found": true,
 *   "description": "I found 'budget_2024.xlsx' from December 18th. Would you like me to res
```

```
 *   "items": [
 *     { "type": "file", "name": "budget_2024.xlsx", "date": "2024-12-18", "id": "..." }
 *   ],
 *   "suggestedActions": ["restore", "download", "show_versions"]
 * }
 */
export async function findInHistory(event: APIGatewayProxyEvent): Promise<APIGatewayProxyRes
  try {
    const chatId = event.pathParameters?.chatId;
    const tenantId = event.requestContext.authorizer?.claims?.['custom:tenant_id'];
    const body = JSON.parse(event.body || '{}');

    if (!chatId || !body.query) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    // Search both messages and files
    const [messages, files] = await Promise.all([
      service.searchMessages(chatId, body.query, tenantId),
      service.searchFiles(chatId, body.query, tenantId),
    ]);

    const items: Array<{
      type: 'message' | 'file';
      name?: string;
      preview?: string;
      date: string;
      id: string;
    }> = [];

    // Add top file results
    for (const file of files.slice(0, 3)) {
      items.push({
        type: 'file',
        name: file.displayName,
        date: file.createdAt.split('T')[0],
        id: file.id,
      });
    }

    // Add top message results
    for (const msg of messages.slice(0, 3)) {
      items.push({
        type: 'message',
        preview: msg.content.substring(0, 100) + (msg.content.length > 100 ? '...' : ''),
        date: msg.createdAt.split('T')[0],
```

```
        id: msg.messageId,
      });
    }

    // Generate description
    let description = '';
    if (items.length === 0) {
      description = `I couldn't find anything matching "${body.query}" in your chat history.
    } else if (files.length > 0 && messages.length === 0) {
      description = `I found ${files.length} file${files.length !== 1 ? 's' : ''} matching '
        `The most recent is '${files[0].displayName}' from ${new Date(files[0].createdAt).to
    } else if (messages.length > 0 && files.length === 0) {
      description = `I found ${messages.length} message${messages.length !== 1 ? 's' : ''} m
    } else {
      description = `I found ${files.length} file${files.length !== 1 ? 's' : ''} and ${mess
        `related to "${body.query}".`;
    }

    const suggestedActions: string[] = [];
    if (files.length > 0) {
      suggestedActions.push('download', 'show_versions');
    }
    if (messages.length > 0) {
      suggestedActions.push('jump_to_message');
    }
    if (items.length > 0) {
      suggestedActions.push('restore');
    }

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({
        found: items.length > 0,
        description,
        items,
        suggestedActions,
      }),
    };
  } catch (error: any) {
    console.error('findInHistory error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

/**
```

```
 * POST /api/ai/chats/:chatId/history/restore
 *
 * AI-assisted restore with natural language confirmation.
 *
 * Request:
 * {
 *   "action": "restore_file",
 *   "fileId": "...",
 *   "confirmed": true
 * }
 *
 * Response:
 * {
 *   "success": true,
 *   "message": "I've restored 'budget_2024.xlsx' to the version from December 18th. Your cu
 *   "undoAvailable": true,
 *   "undoSnapshotId": "..."
 * }
 */
export async function aiRestore(event: APIGatewayProxyEvent): Promise<APIGatewayProxyResult>
  try {
    const chatId = event.pathParameters?.chatId;
    const userId = event.requestContext.authorizer?.claims?.sub;
    const tenantId = event.requestContext.authorizer?.claims?.['custom:tenant_id'];
    const body = JSON.parse(event.body || '{}');

    if (!chatId || !body.action) {
      return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
    }

    // Require confirmation for restore actions
    if (!body.confirmed) {
      let confirmMessage = '';

      switch (body.action) {
        case 'restore_file':
          confirmMessage = "I'll restore this file to the selected version. Your current ver
          break;
        case 'restore_message':
          confirmMessage = "I'll restore this message to how it was at the selected point. C
          break;
        case 'restore_all':
          confirmMessage = "I'll restore the entire conversation to the selected point. All
          break;
        default:
          return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'Unk
```

```
  }

  return {
    statusCode: 200,
    headers: corsHeaders,
    body: JSON.stringify({
      needsConfirmation: true,
      message: confirmMessage,
      action: body.action,
    }),
  };
}

// Perform the restore
let result;
let message = '';

switch (body.action) {
  case 'restore_file':
    result = await service.restore({
      chatId,
      targetSnapshotId: body.snapshotId,
      scope: 'single_file',
      fileIds: [body.fileId],
      reason: 'AI-assisted restore',
    }, userId, tenantId);
    message = `I've restored the file. Your previous version has been saved - snapshot $
    break;

  case 'restore_message':
    result = await service.restore({
      chatId,
      targetSnapshotId: body.snapshotId,
      scope: 'single_message',
      messageIds: [body.messageId],
      reason: 'AI-assisted restore',
    }, userId, tenantId);
    message = `I've restored the message. Your edit history is preserved.`;
    break;

  case 'restore_all':
    result = await service.restore({
      chatId,
      targetSnapshotId: body.snapshotId,
      scope: 'full_chat',
      reason: 'AI-assisted restore',
```

```
        }, userId, tenantId);
        message = `I've restored the conversation to that point. ${result.messagesRestored}
          `Don't worry - everything from before is saved and you can restore it anytime.`;
        break;
    }

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({
        success: true,
        message,
        undoAvailable: true,
        undoSnapshotId: result?.previousSnapshotId,
        newVersion: result?.newVersion,
      }),
    };
  } catch (error: any) {
    console.error('aiRestore error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}

/**
 * GET /api/ai/chats/:chatId/history/compare
 *
 * Compare two points in time - useful for "what changed since..."
 *
 * Query params: from=snapshotId&to=snapshotId (or "current")
 *
 * Response:
 * {
 *   "summary": "Between December 18th and now: 12 new messages, 2 files updated, 1 file ad
 *   "changes": {
 *     "messagesAdded": 12,
 *     "messagesEdited": 3,
 *     "messagesDeleted": 1,
 *     "filesAdded": ["report_v2.xlsx"],
 *     "filesUpdated": ["budget.xlsx", "notes.md"],
 *     "filesDeleted": []
 *   }
 * }
 */
export async function compareSnapshots(event: APIGatewayProxyEvent): Promise<APIGatewayProxy
  try {
    const chatId = event.pathParameters?.chatId;
```

```
const tenantId = event.requestContext.authorizer?.claims?.['custom:tenant_id'];
const fromId = event.queryStringParameters?.from;
const toId = event.queryStringParameters?.to || 'current';

if (!chatId || !fromId) {
  return { statusCode: 400, headers: corsHeaders, body: JSON.stringify({ error: 'chatId
}

// Get states at both points
const fromState = await service.getChatAtSnapshot(chatId, fromId, tenantId);

let toState;
if (toId === 'current') {
  const timeline = await service.getTimeline(chatId, tenantId);
  const currentSnapshotId = timeline.snapshots[timeline.snapshots.length - 1]?.id;
  toState = await service.getChatAtSnapshot(chatId, currentSnapshotId, tenantId);
} else {
  toState = await service.getChatAtSnapshot(chatId, toId, tenantId);
}

// Calculate differences
const fromMessageIds = new Set(fromState.messages.map(m => m.messageId));
const toMessageIds = new Set(toState.messages.map(m => m.messageId));

const messagesAdded = toState.messages.filter(m => !fromMessageIds.has(m.messageId)).len
const messagesRemoved = fromState.messages.filter(m => !toMessageIds.has(m.messageId)).l

const fromFileNames = new Set(fromState.files.map(f => f.originalName));
const toFileNames = new Set(toState.files.map(f => f.originalName));

const filesAdded = toState.files.filter(f => !fromFileNames.has(f.originalName)).map(f =
const filesRemoved = fromState.files.filter(f => !toFileNames.has(f.originalName)).map(f

// Files that exist in both but may have been updated
const filesUpdated = toState.files
  .filter(toFile => {
    const fromFile = fromState.files.find(f => f.originalName === toFile.originalName);
    return fromFile && fromFile.version < toFile.version;
  })
  .map(f => f.originalName);

const fromDate = new Date(fromState.snapshot.timestamp).toLocaleDateString('en-US', {
  month: 'long', day: 'numeric'
});

let summary = `Between ${fromDate} and now: `;
```

```
    const parts: string[] = [];

    if (messagesAdded > 0) parts.push(`${messagesAdded} new message${messagesAdded !== 1 ? '
    if (filesUpdated.length > 0) parts.push(`${filesUpdated.length} file${filesUpdated.lengt
    if (filesAdded.length > 0) parts.push(`${filesAdded.length} file${filesAdded.length !==
    if (messagesRemoved > 0) parts.push(`${messagesRemoved} message${messagesRemoved !== 1 ?

    summary += parts.length > 0 ? parts.join(', ') + '.' : 'no changes.';

    return {
      statusCode: 200,
      headers: corsHeaders,
      body: JSON.stringify({
        summary,
        changes: {
          messagesAdded,
          messagesRemoved,
          filesAdded,
          filesUpdated,
          filesRemoved,
        },
        fromSnapshot: {
          id: fromState.snapshot.id,
          timestamp: fromState.snapshot.timestamp,
          version: fromState.snapshot.version,
        },
        toSnapshot: {
          id: toState.snapshot.id,
          timestamp: toState.snapshot.timestamp,
          version: toState.snapshot.version,
        },
      }),
    };
  } catch (error: any) {
    console.error('compareSnapshots error:', error);
    return { statusCode: 500, headers: corsHeaders, body: JSON.stringify({ error: error.mess
  }
}
```

33.2 AI API Routes

```
// packages/functions/src/routes/ai-time-machine.routes.ts

import { Router } from './router';
```

```typescript
import * as handlers from '../handlers/thinktank/ai-time-machine.handlers';

export function registerAITimeMachineRoutes(router: Router) {
  // Summary and discovery
  router.get('/api/ai/chats/:chatId/history/summary', handlers.getHistorySummary);
  router.post('/api/ai/chats/:chatId/history/find', handlers.findInHistory);

  // AI-assisted restore
  router.post('/api/ai/chats/:chatId/history/restore', handlers.aiRestore);

  // Comparison
  router.get('/api/ai/chats/:chatId/history/compare', handlers.compareSnapshots);
}
```

33.3 Time Machine Visual UI Components

```
// apps/thinktank/src/components/time-machine/time-machine-overlay.tsx

'use client';

import React, { useState, useEffect, useRef } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
import { X, Calendar, Clock, RotateCcw, Download, Search, ChevronLeft, ChevronRight } from '
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { useTimeline, useChatAtSnapshot, useRestore } from '@/hooks/use-time-machine';
import { formatDistanceToNow, format, parseISO, isSameDay } from 'date-fns';
import { cn } from '@/lib/utils';

interface TimeMachineOverlayProps {
  chatId: string;
  isOpen: boolean;
  onClose: () => void;
}

/**
 * Time Machine Overlay
 *
 * Apple Time Machine-inspired visual experience:
 * - Messages stack backwards into "space" with perspective
 * - Timeline bar on the right edge
 * - Calendar picker for jumping to dates
 * - Current state at front, past fading into background
 */
```

```
export function TimeMachineOverlay({ chatId, isOpen, onClose }: TimeMachineOverlayProps) {
  const { data: timeline, isLoading } = useTimeline(chatId);
  const [selectedSnapshotId, setSelectedSnapshotId] = useState<string | null>(null);
  const [viewMode, setViewMode] = useState<'timeline' | 'calendar'>('timeline');
  const [selectedDate, setSelectedDate] = useState<Date>(new Date());

  const { data: snapshotState } = useChatAtSnapshot(chatId, selectedSnapshotId || undefined)
  const restoreMutation = useRestore();

  // Set initial snapshot to latest
  useEffect(() => {
    if (timeline?.snapshots.length && !selectedSnapshotId) {
      setSelectedSnapshotId(timeline.snapshots[timeline.snapshots.length - 1].id);
    }
  }, [timeline, selectedSnapshotId]);

  if (!isOpen) return null;

  const currentIndex = timeline?.snapshots.findIndex(s => s.id === selectedSnapshotId) ?? -1
  const selectedSnapshot = timeline?.snapshots[currentIndex];

  const handleNavigate = (direction: 'prev' | 'next') => {
    if (!timeline) return;

    const newIndex = direction === 'prev'
      ? Math.max(0, currentIndex - 1)
      : Math.min(timeline.snapshots.length - 1, currentIndex + 1);

    setSelectedSnapshotId(timeline.snapshots[newIndex].id);
  };

  const handleRestore = async () => {
    if (!selectedSnapshotId || !timeline) return;

    const isLatest = currentIndex === timeline.snapshots.length - 1;
    if (isLatest) return; // Can't restore to current

    await restoreMutation.mutateAsync({
      chatId,
      snapshotId: selectedSnapshotId,
      scope: 'full_chat',
    });

    onClose();
  };
```

```
return (
  <AnimatePresence>
    <motion.div
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
      exit={{ opacity: 0 }}
      className="fixed inset-0 z-50 bg-black"
    >
      {/* Starfield background (like Time Machine) */}
      <div className="absolute inset-0 overflow-hidden">
        <div className="stars" />
      </div>

      {/* Header */}
      <div className="absolute top-0 left-0 right-0 z-10 p-4 flex items-center justify-bet
        <div className="flex items-center gap-4">
          <Button variant="ghost" size="icon" onClick={onClose} className="text-white">
            <X className="h-6 w-6" />
          </Button>
          <h1 className="text-xl font-semibold text-white">Time Machine</h1>
        </div>

        <div className="flex items-center gap-2">
          <Button
            variant={viewMode === 'timeline' ? 'secondary' : 'ghost'}
            size="sm"
            onClick={() => setViewMode('timeline')}
            className="text-white"
          >
            <Clock className="h-4 w-4 mr-2" />
            Timeline
          </Button>
          <Button
            variant={viewMode === 'calendar' ? 'secondary' : 'ghost'}
            size="sm"
            onClick={() => setViewMode('calendar')}
            className="text-white"
          >
            <Calendar className="h-4 w-4 mr-2" />
            Calendar
          </Button>
        </div>
      </div>

      {/* Main content area */}
      <div className="absolute inset-0 pt-16 pb-24 px-4 flex">
```

```
  {/* Message stack with 3D perspective */}
  <div className="flex-1 relative perspective-1000">
    <MessageStack
      messages={snapshotState?.messages || []}
      files={snapshotState?.files || []}
      isLoading={isLoading}
    />
  </div>

  {/* Right sidebar - Timeline or Calendar */}
  <div className="w-64 ml-4">
    {viewMode === 'timeline' ? (
      <TimelineBar
        snapshots={timeline?.snapshots || []}
        selectedId={selectedSnapshotId}
        onSelect={setSelectedSnapshotId}
      />
    ) : (
      <CalendarPicker
        snapshotsByDate={timeline?.snapshotsByDate || {}}
        selectedDate={selectedDate}
        onSelectDate={(date) => {
          setSelectedDate(date);
          // Find first snapshot on that date
          const dateStr = format(date, 'yyyy-MM-dd');
          const snapshot = timeline?.snapshots.find(s =>
            s.timestamp.startsWith(dateStr)
          );
          if (snapshot) {
            setSelectedSnapshotId(snapshot.id);
          }
        }}
      />
    )}
  </div>
</div>

{/* Bottom control bar */}
<div className="absolute bottom-0 left-0 right-0 p-4 bg-gradient-to-t from-black/80
  <div className="flex items-center justify-between max-w-4xl mx-auto">
    {/* Navigation arrows */}
    <div className="flex items-center gap-2">
      <Button
        variant="outline"
        size="icon"
        onClick={() => handleNavigate('prev')}
```

```
              disabled={currentIndex <= 0}
              className="bg-white/10 border-white/20 text-white"
            >
              <ChevronLeft className="h-4 w-4" />
            </Button>
            <Button
              variant="outline"
              size="icon"
              onClick={() => handleNavigate('next')}
              disabled={currentIndex >= (timeline?.snapshots.length || 0) - 1}
              className="bg-white/10 border-white/20 text-white"
            >
              <ChevronRight className="h-4 w-4" />
            </Button>
          </div>

          {/* Snapshot info */}
          <div className="text-center text-white">
            {selectedSnapshot && (
              <>
                <div className="text-lg font-medium">
                  {format(parseISO(selectedSnapshot.timestamp), 'MMMM d, yyyy')}
                </div>
                <div className="text-sm text-white/60">
                  {format(parseISO(selectedSnapshot.timestamp), 'h:mm a')} ·
                  Version {selectedSnapshot.version} of {timeline?.totalSnapshots}
                </div>
              </>
            )}
          </div>

          {/* Actions */}
          <div className="flex items-center gap-2">
            <Button
              variant="outline"
              className="bg-white/10 border-white/20 text-white"
              disabled={currentIndex === (timeline?.snapshots.length || 0) - 1}
              onClick={handleRestore}
            >
              <RotateCcw className="h-4 w-4 mr-2" />
              Restore
            </Button>
          </div>
        </div>
      </div>
    </motion.div>
```

```
    </AnimatePresence>
  );
}

//
// MESSAGE STACK - 3D perspective view of messages receding into the past
//

function MessageStack({
  messages,
  files,
  isLoading
}: {
  messages: any[];
  files: any[];
  isLoading: boolean;
}) {
  if (isLoading) {
    return (
      <div className="flex items-center justify-center h-full">
        <div className="text-white/60">Loading history...</div>
      </div>
    );
  }

  return (
    <div className="relative h-full overflow-hidden">
      {/* 3D perspective container */}
      <div
        className="absolute inset-0 flex flex-col-reverse items-center justify-end pb-8"
        style={{ transformStyle: 'preserve-3d' }}
      >
        {messages.map((message, index) => {
          // Calculate 3D positioning - newer messages at front, older recede
          const depth = index * 0.5; // How far "back" this message is
          const scale = Math.max(0.3, 1 - depth * 0.1);
          const opacity = Math.max(0.2, 1 - depth * 0.15);
          const blur = Math.min(depth * 0.5, 3);
          const yOffset = index * 60; // Vertical stacking
          const zOffset = -depth * 100; // Depth positioning

          return (
            <motion.div
              key={message.id}
              initial={{ opacity: 0, y: 50 }}
              animate={{ opacity, y: yOffset }}
```

```
              className={cn(
                "w-full max-w-2xl p-4 rounded-lg mb-2",
                message.role === 'user'
                  ? 'bg-blue-600/80 ml-auto mr-0'
                  : 'bg-gray-700/80 mr-auto ml-0'
              )}
              style={{
                transform: `scale(${scale}) translateZ(${zOffset}px)`,
                filter: `blur(${blur}px)`,
              }}
            >
              <div className="text-sm text-white/60 mb-1">
                {message.role === 'user' ? 'You' : 'AI'}
              </div>
              <div className="text-white">
                {message.content.length > 200
                  ? message.content.substring(0, 200) + '...'
                  : message.content}
              </div>
            </motion.div>
          );
        })}
      </div>

      {/* Files bar at bottom */}
      {files.length > 0 && (
        <div className="absolute bottom-0 left-0 right-0 p-4 bg-gradient-to-t from-black/60'
          <div className="flex gap-2 overflow-x-auto">
            {files.map(file => (
              <div
                key={file.id}
                className="flex-shrink-0 px-3 py-2 bg-white/10 rounded-lg text-white text-sm
              >
                  {file.displayName}
              </div>
            ))}
          </div>
        </div>
      )}
    </div>
  );
}

//
// TIMELINE BAR - Visual timeline of snapshots
//
```

```
function TimelineBar({
  snapshots,
  selectedId,
  onSelect,
}: {
  snapshots: any[];
  selectedId: string | null;
  onSelect: (id: string) => void;
}) {
  const containerRef = useRef<HTMLDivElement>(null);

  // Group snapshots by date
  const groupedByDate: Record<string, any[]> = {};
  for (const snapshot of snapshots) {
    const date = snapshot.timestamp.split('T')[0];
    if (!groupedByDate[date]) {
      groupedByDate[date] = [];
    }
    groupedByDate[date].push(snapshot);
  }

  return (
    <div ref={containerRef} className="h-full overflow-y-auto pr-2">
      {Object.entries(groupedByDate).reverse().map(([date, dateSnapshots]) => (
        <div key={date} className="mb-4">
          <div className="text-xs text-white/40 mb-2 sticky top-0 bg-black/50 py-1">
            {format(parseISO(date), 'MMM d, yyyy')}
          </div>
          <div className="space-y-1">
            {dateSnapshots.map(snapshot => (
              <button
                key={snapshot.id}
                onClick={() => onSelect(snapshot.id)}
                className={cn(
                  "w-full text-left px-3 py-2 rounded-lg transition-colors",
                  "text-sm text-white/80 hover:bg-white/10",
                  selectedId === snapshot.id && "bg-white/20 ring-1 ring-white/40"
                )}
              >
                <div className="font-medium">
                  {format(parseISO(snapshot.timestamp), 'h:mm a')}
                </div>
                <div className="text-xs text-white/50">
                  {snapshot.messageCount} messages · {snapshot.fileCount} files
                </div>
```

```
              </button>
            ))}
          </div>
        </div>
      ))}
    </div>
  );
}

//
// CALENDAR PICKER - Jump to any date
//

function CalendarPicker({
  snapshotsByDate,
  selectedDate,
  onSelectDate,
}: {
  snapshotsByDate: Record<string, number>;
  selectedDate: Date;
  onSelectDate: (date: Date) => void;
}) {
  const [viewMonth, setViewMonth] = useState(selectedDate);

  // Generate calendar grid
  const daysInMonth = new Date(viewMonth.getFullYear(), viewMonth.getMonth() + 1, 0).getDate
  const firstDayOfMonth = new Date(viewMonth.getFullYear(), viewMonth.getMonth(), 1).getDay(

  const days: (number | null)[] = [];
  for (let i = 0; i < firstDayOfMonth; i++) {
    days.push(null);
  }
  for (let i = 1; i <= daysInMonth; i++) {
    days.push(i);
  }

  return (
    <div className="bg-white/5 rounded-lg p-4">
      {/* Month navigation */}
      <div className="flex items-center justify-between mb-4">
        <Button
          variant="ghost"
          size="icon"
          onClick={() => setViewMonth(new Date(viewMonth.getFullYear(), viewMonth.getMonth()
          className="text-white"
        >
```

```
    <ChevronLeft className="h-4 w-4" />
  </Button>
  <div className="text-white font-medium">
    {format(viewMonth, 'MMMM yyyy')}
  </div>
  <Button
    variant="ghost"
    size="icon"
    onClick={() => setViewMonth(new Date(viewMonth.getFullYear(), viewMonth.getMonth()
    className="text-white"
  >
    <ChevronRight className="h-4 w-4" />
  </Button>
</div>

{/* Day labels */}
<div className="grid grid-cols-7 gap-1 mb-2">
  {['S', 'M', 'T', 'W', 'T', 'F', 'S'].map((day, i) => (
    <div key={i} className="text-center text-xs text-white/40">
      {day}
    </div>
  ))}
</div>

{/* Calendar grid */}
<div className="grid grid-cols-7 gap-1">
  {days.map((day, i) => {
    if (day === null) {
      return <div key={i} />;
    }

    const date = new Date(viewMonth.getFullYear(), viewMonth.getMonth(), day);
    const dateStr = format(date, 'yyyy-MM-dd');
    const hasSnapshots = snapshotsByDate[dateStr] > 0;
    const isSelected = isSameDay(date, selectedDate);
    const isToday = isSameDay(date, new Date());

    return (
      <button
        key={i}
        onClick={() => hasSnapshots && onSelectDate(date)}
        disabled={!hasSnapshots}
        className={cn(
          "aspect-square flex items-center justify-center rounded-full text-sm",
          "transition-colors",
          hasSnapshots
```

```
                ? "text-white hover:bg-white/20 cursor-pointer"
                : "text-white/20 cursor-not-allowed",
              isSelected && "bg-blue-600 text-white",
              isToday && !isSelected && "ring-1 ring-white/40"
            )}
          >
            {day}
            {hasSnapshots && !isSelected && (
              <span className="absolute bottom-0.5 w-1 h-1 bg-blue-400 rounded-full" />
            )}
          </button>
        );
      })}
    </div>
  </div>
);
}
```

33.4 Time Machine Entry Button

```
// apps/thinktank/src/components/time-machine/time-machine-button.tsx

'use client';

import React, { useState } from 'react';
import { Clock } from 'lucide-react';
import { Button } from '@/components/ui/button';
import { Tooltip, TooltipContent, TooltipTrigger } from '@/components/ui/tooltip';
import { TimeMachineOverlay } from './time-machine-overlay';

interface TimeMachineButtonProps {
  chatId: string;
}

/**
 * Time Machine Entry Button
 *
 * Small, unobtrusive button that opens the full Time Machine experience.
 * Hidden in the chat header, only visible on hover or in Advanced mode.
 */
export function TimeMachineButton({ chatId }: TimeMachineButtonProps) {
  const [isOpen, setIsOpen] = useState(false);

  return (
```

```
    <>
      <Tooltip>
        <TooltipTrigger asChild>
          <Button
            variant="ghost"
            size="sm"
            onClick={() => setIsOpen(true)}
            className="h-8 px-2 text-muted-foreground hover:text-foreground"
          >
            <Clock className="h-4 w-4 mr-1.5" />
            <span className="text-xs">Time Machine</span>
          </Button>
        </TooltipTrigger>
        <TooltipContent>
          <p>Browse and restore chat history</p>
        </TooltipContent>
      </Tooltip>

      <TimeMachineOverlay
        chatId={chatId}
        isOpen={isOpen}
        onClose={() => setIsOpen(false)}
      />
    </>
  );
}
```

33.5 React Hooks for Time Machine

```
// apps/thinktank/src/hooks/use-time-machine.ts

import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { api } from '@/lib/api';
import type { ChatTimeline, RestoreResult, RestoreScope } from '@radiant/shared';

// Get full timeline
export function useTimeline(chatId: string | undefined) {
  return useQuery({
    queryKey: ['time-machine', 'timeline', chatId],
    queryFn: async () => {
      const response = await api.get<ChatTimeline>(`/thinktank/chats/${chatId}/time-machine
      return response.data;
    },
    enabled: !!chatId,
```

```
    staleTime: 30000, // 30 seconds
  });
}

// Get chat state at specific snapshot
export function useChatAtSnapshot(chatId: string | undefined, snapshotId: string | undefined
  return useQuery({
    queryKey: ['time-machine', 'snapshot', chatId, snapshotId],
    queryFn: async () => {
      const response = await api.get(`/thinktank/chats/${chatId}/time-machine/snapshots/${sn
      return response.data;
    },
    enabled: !!chatId && !!snapshotId,
  });
}

// Get snapshots for a specific date
export function useSnapshotsByDate(chatId: string | undefined, date: string | undefined) {
  return useQuery({
    queryKey: ['time-machine', 'date', chatId, date],
    queryFn: async () => {
      const response = await api.get(`/thinktank/chats/${chatId}/time-machine/calendar/${dat
      return response.data.snapshots;
    },
    enabled: !!chatId && !!date,
  });
}

// Restore mutation
export function useRestore() {
  const queryClient = useQueryClient();

  return useMutation({
    mutationFn: async ({
      chatId,
      snapshotId,
      scope = 'full_chat',
      messageIds,
      fileIds,
      reason,
    }: {
      chatId: string;
      snapshotId: string;
      scope?: RestoreScope;
      messageIds?: string[];
      fileIds?: string[];
```

```
      reason?: string;
    }) => {
      const response = await api.post<RestoreResult>(`/thinktank/chats/${chatId}/time-machin
        snapshotId,
        scope,
        messageIds,
        fileIds,
        reason,
      });
      return response.data;
    },
    onSuccess: (_, variables) => {
      // Invalidate all related queries
      queryClient.invalidateQueries({ queryKey: ['time-machine', 'timeline', variables.chatI
      queryClient.invalidateQueries({ queryKey: ['chat', variables.chatId] });
      queryClient.invalidateQueries({ queryKey: ['messages', variables.chatId] });
    },
  });
}

// Search in history
export function useHistorySearch(chatId: string | undefined, query: string) {
  return useQuery({
    queryKey: ['time-machine', 'search', chatId, query],
    queryFn: async () => {
      const response = await api.get(`/thinktank/chats/${chatId}/time-machine/search`, {
        params: { q: query },
      });
      return response.data;
    },
    enabled: !!chatId && query.length > 2,
  });
}

// File versions
export function useFileVersions(chatId: string | undefined, fileName: string | undefined) {
  return useQuery({
    queryKey: ['time-machine', 'file-versions', chatId, fileName],
    queryFn: async () => {
      const response = await api.get(`/thinktank/chats/${chatId}/time-machine/files/${encode
      return response.data.versions;
    },
    enabled: !!chatId && !!fileName,
  });
}
```

```typescript
// Create export
export function useCreateExport() {
  return useMutation({
    mutationFn: async ({
      chatId,
      format = 'zip',
      includeMedia = true,
      includeVersionHistory = false,
    }: {
      chatId: string;
      format?: 'zip' | 'json' | 'markdown' | 'pdf' | 'html';
      includeMedia?: boolean;
      includeVersionHistory?: boolean;
    }) => {
      const response = await api.post(`/thinktank/chats/${chatId}/time-machine/export`, {
        format,
        includeMedia,
        includeVersionHistory,
      });
      return response.data;
    },
  });
}
```

33.6 CDK Infrastructure Updates

```typescript
// packages/cdk/src/stacks/time-machine-stack.ts

import * as cdk from 'aws-cdk-lib';
import * as s3 from 'aws-cdk-lib/aws-s3';
import * as iam from 'aws-cdk-lib/aws-iam';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as apigateway from 'aws-cdk-lib/aws-apigateway';
import { Construct } from 'constructs';

interface TimeMachineStackProps extends cdk.StackProps {
  environment: string;
  thinktankApi: apigateway.RestApi;
  thinktankLambda: lambda.Function;
}

export class TimeMachineStack extends cdk.Stack {
  public readonly mediaVaultBucket: s3.Bucket;
```

```typescript
constructor(scope: Construct, id: string, props: TimeMachineStackProps) {
  super(scope, id, props);

  //
  // MEDIA VAULT BUCKET - S3 with versioning, never delete
  //

  this.mediaVaultBucket = new s3.Bucket(this, 'MediaVault', {
    bucketName: `radiant-media-vault-${props.environment}-${cdk.Aws.ACCOUNT_ID}`,

    // CRITICAL: Enable versioning for true immutability
    versioned: true,

    // Security
    blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
    encryption: s3.BucketEncryption.S3_MANAGED,

    // CORS for direct uploads
    cors: [{
      allowedHeaders: ['*'],
      allowedMethods: [s3.HttpMethods.PUT, s3.HttpMethods.POST, s3.HttpMethods.GET],
      allowedOrigins: ['*'],
      exposedHeaders: ['ETag', 'x-amz-version-id'],
      maxAge: 3600,
    }],

    // Lifecycle: Move to cheaper storage, NEVER delete
    lifecycleRules: [{
      id: 'TransitionToIntelligentTiering',
      transitions: [{
        storageClass: s3.StorageClass.INTELLIGENT_TIERING,
        transitionAfter: cdk.Duration.days(30),
      }],
      noncurrentVersionTransitions: [{
        storageClass: s3.StorageClass.GLACIER_INSTANT_RETRIEVAL,
        transitionAfter: cdk.Duration.days(90),
      }],
      // NO expiration - keep forever
    }],

    // Transfer acceleration
    transferAcceleration: true,

    // RETAIN even if stack deleted
    removalPolicy: cdk.RemovalPolicy.RETAIN,
  });
```

```
// Deny delete actions (belt and suspenders)
this.mediaVaultBucket.addToResourcePolicy(new iam.PolicyStatement({
  sid: 'DenyObjectDeletion',
  effect: iam.Effect.DENY,
  principals: [new iam.AnyPrincipal()],
  actions: ['s3:DeleteObject', 's3:DeleteObjectVersion'],
  resources: [this.mediaVaultBucket.arnForObjects('*')],
  conditions: {
    StringNotEquals: {
      'aws:PrincipalArn': [
        `arn:aws:iam::${cdk.Aws.ACCOUNT_ID}:role/radiant-gdpr-deletion-role`,
      ],
    },
  },
}));

// Grant Lambda read/write (but not delete)
this.mediaVaultBucket.grantReadWrite(props.thinktankLambda);
props.thinktankLambda.addEnvironment('MEDIA_VAULT_BUCKET', this.mediaVaultBucket.bucketN

//
// API ROUTES
//

const api = props.thinktankApi;
const lambdaIntegration = new apigateway.LambdaIntegration(props.thinktankLambda);

// Time Machine base
const chatsResource = api.root.addResource('chats');
const chatResource = chatsResource.addResource('{chatId}');
const timeMachineResource = chatResource.addResource('time-machine');

// Timeline
timeMachineResource.addMethod('GET', lambdaIntegration);

// Snapshots
const snapshotsResource = timeMachineResource.addResource('snapshots');
snapshotsResource.addResource('{snapshotId}').addMethod('GET', lambdaIntegration);

// Calendar
timeMachineResource.addResource('calendar').addResource('{date}').addMethod('GET', lambd

// Restore
timeMachineResource.addResource('restore').addMethod('POST', lambdaIntegration);
```

```
    // Files
    const filesResource = timeMachineResource.addResource('files');
    filesResource.addMethod('GET', lambdaIntegration);
    filesResource.addResource('{fileName}').addResource('versions').addMethod('GET', lambdaI

    // Search
    timeMachineResource.addResource('search').addMethod('GET', lambdaIntegration);

    // Export
    timeMachineResource.addResource('export').addMethod('POST', lambdaIntegration);

    // File download
    api.root.addResource('files').addResource('{fileId}').addResource('download').addMethod(

    // Export status
    api.root.addResource('exports').addResource('{bundleId}').addMethod('GET', lambdaIntegra

    //
    // AI API ROUTES
    //

    const aiResource = api.root.addResource('ai');
    const aiChatsResource = aiResource.addResource('chats').addResource('{chatId}');
    const historyResource = aiChatsResource.addResource('history');

    historyResource.addResource('summary').addMethod('GET', lambdaIntegration);
    historyResource.addResource('find').addMethod('POST', lambdaIntegration);
    historyResource.addResource('restore').addMethod('POST', lambdaIntegration);
    historyResource.addResource('compare').addMethod('GET', lambdaIntegration);

    //
    // OUTPUTS
    //

    new cdk.CfnOutput(this, 'MediaVaultBucketName', {
      value: this.mediaVaultBucket.bucketName,
      description: 'Time Machine Media Vault S3 Bucket',
    });
  }
}
```

33.7 Integration with Think Tank Chat

```
// apps/thinktank/src/components/chat/chat-header.tsx (updated)

import { TimeMachineButton } from '../time-machine/time-machine-button';

export function ChatHeader({ chatId, title }: ChatHeaderProps) {
  return (
    <div className="flex items-center justify-between p-4 border-b">
      <div className="flex items-center gap-2">
        <h1 className="font-semibold">{title || 'New Chat'}</h1>
      </div>
      <div className="flex items-center gap-2">
        {/* Time Machine - hidden until hover */}
        <div className="opacity-0 group-hover:opacity-100 transition-opacity">
          <TimeMachineButton chatId={chatId} />
        </div>

        {/* Other header actions */}
        <Button variant="ghost" size="sm">
          <MoreHorizontal className="h-4 w-4" />
        </Button>
      </div>
    </div>
  );
}
```

SECTION-34-ORCHESTRATION-ENGINE

SECTION 34: DATABASE-DRIVEN ORCHESTRATION ENGINE (v4.1.0)

CRITICAL: This section replaces ALL hardcoded model configurations with database-driven management. All model configs, workflows, and orchestration parameters are stored in PostgreSQL. Administrators can add/edit/delete models entirely through the Admin Dashboard UI.

34.1 DATABASE SCHEMA

packages/database/migrations/034_orchestration_engine.sql

```sql
-- ============================================================================
-- RADIANT v4.1.0 - DATABASE-DRIVEN ORCHESTRATION ENGINE
-- ============================================================================
-- This migration creates the foundation for dynamic model management.
-- ALL model configurations are stored in PostgreSQL - NO HARDCODING.
-- ============================================================================

-- AI Model Registry (Replaces hardcoded TypeScript configs)
CREATE TABLE IF NOT EXISTS ai_models (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  model_id VARCHAR(100) UNIQUE NOT NULL,
  name VARCHAR(100) NOT NULL,
  display_name VARCHAR(200) NOT NULL,
  description TEXT,
  category VARCHAR(50) NOT NULL,
  specialty VARCHAR(50),
  provider_type VARCHAR(20) NOT NULL DEFAULT 'self_hosted',

  deployment_config JSONB NOT NULL DEFAULT '{}',
  parameters BIGINT DEFAULT 0,
  accuracy VARCHAR(200),
  benchmark TEXT,
  capabilities TEXT[] DEFAULT '{}',
  input_formats TEXT[] DEFAULT '{}',
  output_formats TEXT[] DEFAULT '{}',
  architecture JSONB DEFAULT '{}',
  performance_metrics JSONB DEFAULT '{}',

  thermal_config JSONB NOT NULL DEFAULT '{"defaultState":"OFF","scaleToZeroAfterMinutes":15,
  current_thermal_state VARCHAR(20) DEFAULT 'OFF',
  last_thermal_change TIMESTAMPTZ,

  pricing_config JSONB NOT NULL DEFAULT '{"hourlyRate":0,"perRequest":0,"markup":0.75}',

  min_tier INTEGER DEFAULT 1,
  requires_gpu BOOLEAN DEFAULT false,
  gpu_memory_gb INTEGER DEFAULT 0,
  enabled BOOLEAN DEFAULT true,
  status VARCHAR(20) DEFAULT 'active',
  version VARCHAR(50),
  repository VARCHAR(500),
  release_date DATE,
```

```
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  created_by UUID REFERENCES administrators(id),
  updated_by UUID REFERENCES administrators(id)
);

-- License Tracking
CREATE TABLE IF NOT EXISTS model_licenses (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  model_id UUID NOT NULL REFERENCES ai_models(id) ON DELETE CASCADE,
  license_type VARCHAR(50) NOT NULL,
  license_spdx VARCHAR(50) NOT NULL,
  license_url VARCHAR(500),
  commercial_use BOOLEAN DEFAULT true,
  commercial_notes TEXT,
  attribution_required BOOLEAN DEFAULT false,
  attribution_text TEXT,
  share_alike BOOLEAN DEFAULT false,
  compliance_status VARCHAR(20) DEFAULT 'compliant',
  last_compliance_review TIMESTAMPTZ,
  reviewed_by UUID REFERENCES administrators(id),
  compliance_notes TEXT,
  expires_at TIMESTAMPTZ,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Model Dependencies
CREATE TABLE IF NOT EXISTS model_dependencies (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  model_id UUID NOT NULL REFERENCES ai_models(id) ON DELETE CASCADE,
  dependency_type VARCHAR(50) NOT NULL,
  dependency_name VARCHAR(200) NOT NULL,
  dependency_size_gb DECIMAL(10,2),
  dependency_license VARCHAR(50),
  required BOOLEAN DEFAULT true,
  notes TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Workflow Definitions
CREATE TABLE IF NOT EXISTS workflow_definitions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  workflow_id VARCHAR(100) UNIQUE NOT NULL,
  name VARCHAR(200) NOT NULL,
```

```
  description TEXT,
  category VARCHAR(50) NOT NULL,
  version VARCHAR(20) NOT NULL DEFAULT '1.0.0',
  dag_definition JSONB NOT NULL,
  input_schema JSONB NOT NULL DEFAULT '{}',
  output_schema JSONB NOT NULL DEFAULT '{}',
  default_parameters JSONB NOT NULL DEFAULT '{}',
  timeout_seconds INTEGER DEFAULT 3600,
  max_retries INTEGER DEFAULT 3,
  min_tier INTEGER DEFAULT 1,
  enabled BOOLEAN DEFAULT true,
  requires_audit_trail BOOLEAN DEFAULT false,
  hipaa_compliant BOOLEAN DEFAULT false,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  created_by UUID REFERENCES administrators(id)
);

-- Workflow Tasks
CREATE TABLE IF NOT EXISTS workflow_tasks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  workflow_id UUID NOT NULL REFERENCES workflow_definitions(id) ON DELETE CASCADE,
  task_id VARCHAR(100) NOT NULL,
  name VARCHAR(200) NOT NULL,
  description TEXT,
  task_type VARCHAR(50) NOT NULL,
  model_id UUID REFERENCES ai_models(id),
  service_id VARCHAR(100),
  config JSONB NOT NULL DEFAULT '{}',
  input_mapping JSONB DEFAULT '{}',
  output_mapping JSONB DEFAULT '{}',
  sequence_order INTEGER DEFAULT 0,
  depends_on TEXT[] DEFAULT '{}',
  condition_expression TEXT,
  timeout_seconds INTEGER DEFAULT 300,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(workflow_id, task_id)
);

-- Workflow Parameters
CREATE TABLE IF NOT EXISTS workflow_parameters (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  workflow_id UUID NOT NULL REFERENCES workflow_definitions(id) ON DELETE CASCADE,
  parameter_name VARCHAR(100) NOT NULL,
  display_name VARCHAR(200),
```

```
  description TEXT,
  data_type VARCHAR(50) NOT NULL,
  default_value JSONB,
  validation_rules JSONB DEFAULT '{}',
  ui_component VARCHAR(50) DEFAULT 'text',
  ui_config JSONB DEFAULT '{}',
  user_configurable BOOLEAN DEFAULT true,
  admin_only BOOLEAN DEFAULT false,
  sequence_order INTEGER DEFAULT 0,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(workflow_id, parameter_name)
);

-- Workflow Executions
CREATE TABLE IF NOT EXISTS workflow_executions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  workflow_id UUID NOT NULL REFERENCES workflow_definitions(id),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  user_id UUID NOT NULL REFERENCES users(id),
  status VARCHAR(20) NOT NULL DEFAULT 'pending',
  input_parameters JSONB NOT NULL DEFAULT '{}',
  resolved_parameters JSONB DEFAULT '{}',
  output_data JSONB,
  error_message TEXT,
  error_details JSONB,
  started_at TIMESTAMPTZ,
  completed_at TIMESTAMPTZ,
  duration_ms INTEGER,
  estimated_cost_usd DECIMAL(10,4),
  actual_cost_usd DECIMAL(10,4),
  checkpoint_data JSONB,
  priority INTEGER DEFAULT 5,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Task Executions
CREATE TABLE IF NOT EXISTS task_executions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  workflow_execution_id UUID NOT NULL REFERENCES workflow_executions(id) ON DELETE CASCADE,
  task_id VARCHAR(100) NOT NULL,
  status VARCHAR(20) NOT NULL DEFAULT 'pending',
  attempt_number INTEGER DEFAULT 1,
  input_data JSONB,
  output_data JSONB,
  error_message TEXT,
```

```
  error_code VARCHAR(50),
  started_at TIMESTAMPTZ,
  completed_at TIMESTAMPTZ,
  duration_ms INTEGER,
  resource_usage JSONB DEFAULT '{}',
  cost_usd DECIMAL(10,4),
  model_id UUID REFERENCES ai_models(id),
  model_endpoint VARCHAR(500),
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Orchestration Audit Log
CREATE TABLE IF NOT EXISTS orchestration_audit_log (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  entity_type VARCHAR(50) NOT NULL,
  entity_id UUID NOT NULL,
  action VARCHAR(50) NOT NULL,
  actor_id UUID REFERENCES administrators(id),
  actor_type VARCHAR(20) DEFAULT 'admin',
  previous_state JSONB,
  new_state JSONB,
  change_summary TEXT,
  ip_address INET,
  user_agent TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Indexes
CREATE INDEX IF NOT EXISTS idx_ai_models_category ON ai_models(category);
CREATE INDEX IF NOT EXISTS idx_ai_models_status ON ai_models(status, enabled);
CREATE INDEX IF NOT EXISTS idx_model_licenses_model ON model_licenses(model_id);
CREATE INDEX IF NOT EXISTS idx_model_licenses_compliance ON model_licenses(compliance_status
CREATE INDEX IF NOT EXISTS idx_workflow_executions_tenant ON workflow_executions(tenant_id);
CREATE INDEX IF NOT EXISTS idx_workflow_executions_status ON workflow_executions(status);
CREATE INDEX IF NOT EXISTS idx_orchestration_audit_time ON orchestration_audit_log(created_a

-- Function: Get full model config (replaces hardcoded lookups)
CREATE OR REPLACE FUNCTION get_model_config(p_model_id VARCHAR)
RETURNS JSONB AS $$
DECLARE
  result JSONB;
BEGIN
  SELECT jsonb_build_object(
    'id', m.model_id,
    'name', m.name,
```

```
    'displayName', m.display_name,
    'description', m.description,
    'category', m.category,
    'providerType', m.provider_type,
    'deployment', m.deployment_config,
    'thermal', m.thermal_config,
    'pricing', m.pricing_config,
    'parameters', m.parameters,
    'accuracy', m.accuracy,
    'minTier', m.min_tier,
    'requiresGpu', m.requires_gpu,
    'status', m.status,
    'version', m.version,
    'currentThermalState', m.current_thermal_state,
    'licenses', (
      SELECT COALESCE(jsonb_agg(jsonb_build_object(
        'id', l.id, 'type', l.license_type, 'spdx', l.license_spdx,
        'commercialUse', l.commercial_use, 'complianceStatus', l.compliance_status
      )), '[]'::jsonb)
      FROM model_licenses l WHERE l.model_id = m.id
    ),
    'dependencies', (
      SELECT COALESCE(jsonb_agg(jsonb_build_object(
        'name', d.dependency_name, 'type', d.dependency_type,
        'sizeGb', d.dependency_size_gb, 'required', d.required
      )), '[]'::jsonb)
      FROM model_dependencies d WHERE d.model_id = m.id
    )
  ) INTO result
  FROM ai_models m
  WHERE m.model_id = p_model_id AND m.enabled = true;

  RETURN result;
END;
$$ LANGUAGE plpgsql STABLE;

-- Function: Get workflow config
CREATE OR REPLACE FUNCTION get_workflow_config(p_workflow_id VARCHAR)
RETURNS JSONB AS $$
DECLARE
  result JSONB;
BEGIN
  SELECT jsonb_build_object(
    'id', w.workflow_id,
    'name', w.name,
    'description', w.description,
```

```
    'category', w.category,
    'version', w.version,
    'dag', w.dag_definition,
    'inputSchema', w.input_schema,
    'defaultParameters', w.default_parameters,
    'timeout', w.timeout_seconds,
    'minTier', w.min_tier,
    'tasks', (
      SELECT COALESCE(jsonb_agg(jsonb_build_object(
        'taskId', t.task_id, 'name', t.name, 'type', t.task_type,
        'modelId', (SELECT model_id FROM ai_models WHERE id = t.model_id),
        'config', t.config, 'dependsOn', t.depends_on, 'timeout', t.timeout_seconds
      ) ORDER BY t.sequence_order), '[]'::jsonb)
      FROM workflow_tasks t WHERE t.workflow_id = w.id
    ),
    'parameters', (
      SELECT COALESCE(jsonb_agg(jsonb_build_object(
        'name', p.parameter_name, 'displayName', p.display_name,
        'type', p.data_type, 'default', p.default_value,
        'validation', p.validation_rules, 'uiComponent', p.ui_component
      ) ORDER BY p.sequence_order), '[]'::jsonb)
      FROM workflow_parameters p WHERE p.workflow_id = w.id
    )
  ) INTO result
  FROM workflow_definitions w
  WHERE w.workflow_id = p_workflow_id AND w.enabled = true;

  RETURN result;
END;
$$ LANGUAGE plpgsql STABLE;

-- Audit Trigger
CREATE OR REPLACE FUNCTION log_model_changes()
RETURNS TRIGGER AS $$
BEGIN
  IF TG_OP = 'INSERT' THEN
    INSERT INTO orchestration_audit_log (entity_type, entity_id, action, new_state, change_s
    VALUES ('model', NEW.id, 'created', to_jsonb(NEW), 'Model ' || NEW.display_name || ' cre
  ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO orchestration_audit_log (entity_type, entity_id, action, previous_state, new
    VALUES ('model', NEW.id, 'updated', to_jsonb(OLD), to_jsonb(NEW), 'Model ' || NEW.displa
  ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO orchestration_audit_log (entity_type, entity_id, action, previous_state, cha
    VALUES ('model', OLD.id, 'deleted', to_jsonb(OLD), 'Model ' || OLD.display_name || ' del
  END IF;
  RETURN COALESCE(NEW, OLD);
```

```
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS ai_models_audit_trigger ON ai_models;
CREATE TRIGGER ai_models_audit_trigger
AFTER INSERT OR UPDATE OR DELETE ON ai_models
FOR EACH ROW EXECUTE FUNCTION log_model_changes();

-- Row Level Security
ALTER TABLE ai_models ENABLE ROW LEVEL SECURITY;
ALTER TABLE workflow_executions ENABLE ROW LEVEL SECURITY;

CREATE POLICY ai_models_read ON ai_models FOR SELECT TO authenticated
  USING (enabled = true AND status IN ('active', 'beta'));

CREATE POLICY workflow_executions_owner ON workflow_executions FOR ALL TO authenticated
  USING (user_id = auth.uid() OR EXISTS (SELECT 1 FROM administrators WHERE user_id = auth.u
```

34.2 ALPHAFOLD 2 SEED DATA

packages/database/migrations/034a_seed_alphafold2.sql

```
-- ============================================================================
-- ALPHAFOLD 2 - Nobel Prize-Winning Protein Folding Model
-- ============================================================================

INSERT INTO ai_models (
  model_id, name, display_name, description, category, specialty, provider_type,
  deployment_config, parameters, accuracy, benchmark, capabilities,
  input_formats, output_formats, architecture, performance_metrics,
  thermal_config, pricing_config, min_tier, requires_gpu, gpu_memory_gb,
  status, version, repository, release_date
) VALUES (
  'alphafold2',
  'alphafold2',
  'AlphaFold 2',
  'Nobel Prize-winning protein structure prediction with near-experimental accuracy. Won CAS
  'scientific_protein',
  'protein_folding',
  'self_hosted',
  '{
    "image": "pytorch-inference:2.1-gpu-py310-cu121-ubuntu22.04",
    "instanceType": "ml.g5.12xlarge",
    "environment": {"MODEL_NAME": "alphafold2_ptm", "JAX_PLATFORM_NAME": "gpu"},
    "modelDataUrl": "s3://radiant-models/alphafold2/params_2022-12-06.tar",
```

```
    "resourceRequirements": {"minGpuMemoryGB": 16, "recommendedGpuMemoryGB": 96, "databaseSt
  }'::JSONB,
  93000000,
  'GDT > 90 on CASP14, 92.4 median GDT_TS, 0.96Å median backbone RMSD',
  'CASP14: Won 88/97 targets, far exceeding all other methods.',
  ARRAY['protein_folding', 'structure_prediction', 'confidence_estimation', 'multimer_predic
  ARRAY['text/fasta', 'text/plain', 'application/json'],
  ARRAY['application/pdb', 'application/mmcif', 'application/json'],
  '{"evoformerBlocks": 48, "structureBlocks": 8, "recyclingIterations": 4}'::JSONB,
  '{"inferenceTime100Residues": 4.9, "inferenceTime500Residues": 29, "maxResiduesG5_12xlarge
  '{"defaultState": "OFF", "scaleToZeroAfterMinutes": 10, "warmupTimeSeconds": 180, "minInst
  '{"hourlyRate": 14.28, "perRequest": 2.00, "perResidueOver500": 0.002, "markup": 0.75}'::J
  4, true, 96, 'active', '2.3.2', 'https://github.com/google-deepmind/alphafold', '2023-04-0
) ON CONFLICT (model_id) DO UPDATE SET updated_at = NOW();

-- Insert licenses
DO $$
DECLARE alphafold2_id UUID;
BEGIN
  SELECT id INTO alphafold2_id FROM ai_models WHERE model_id = 'alphafold2';

  INSERT INTO model_licenses (model_id, license_type, license_spdx, license_url, commercial_
  VALUES
    (alphafold2_id, 'source_code', 'Apache-2.0', 'https://github.com/google-deepmind/alphafo
    (alphafold2_id, 'model_weights', 'CC-BY-4.0', 'https://creativecommons.org/licenses/by/4
    (alphafold2_id, 'database', 'CC-BY-SA-4.0', 'https://creativecommons.org/licenses/by-sa/
  ON CONFLICT DO NOTHING;

  INSERT INTO model_dependencies (model_id, dependency_type, dependency_name, dependency_siz
  VALUES
    (alphafold2_id, 'database', 'BFD', 1800, 'CC-BY-SA-4.0', true),
    (alphafold2_id, 'database', 'UniRef90', 103, 'CC-BY-4.0', true),
    (alphafold2_id, 'database', 'MGnify', 120, 'CC0-1.0', true),
    (alphafold2_id, 'database', 'PDB70', 56, 'CC0-1.0', true),
    (alphafold2_id, 'database', 'PDB mmCIF', 238, 'CC0-1.0', true)
  ON CONFLICT DO NOTHING;
END $$;

-- Insert protein folding workflow
INSERT INTO workflow_definitions (
  workflow_id, name, description, category, version, dag_definition, input_schema, output_sc
  default_parameters, timeout_seconds, max_retries, min_tier, requires_audit_trail
) VALUES (
  'protein_folding_alphafold2',
  'AlphaFold 2 Protein Folding Pipeline',
  'Complete protein structure prediction using AlphaFold 2 with MSA generation, template sea
```

```
  'scientific',
  '1.0.0',
  '{"nodes":[{"id":"validate_input","type":"validation"},{"id":"msa_generation","type":"task
  '{"type":"object","required":["sequence"],"properties":{"sequence":{"type":"string","minLe
  '{"type":"object","properties":{"structures":{"type":"array"},"confidence":{"type":"object
  '{"numModels":5,"recyclingIterations":4,"relaxStructure":true,"useTemplates":true}'::JSONB
  7200, 2, 4, true
) ON CONFLICT (workflow_id) DO UPDATE SET updated_at = NOW();
```

34.3 ORCHESTRATION ENGINE SERVICE

packages/services/orchestration/OrchestrationEngine.ts

```typescript
/**
 * RADIANT v4.1.0 - Database-Driven Orchestration Engine
 *
 * KEY PRINCIPLE: ZERO HARDCODING
 * All model configs stored in ai_models table - retrieved via get_model_config()
 */

import { Pool } from 'pg';
import { EventEmitter } from 'events';

export interface ModelConfig {
  id: string;
  name: string;
  displayName: string;
  description?: string;
  category: string;
  providerType: 'self_hosted' | 'external';
  deployment: Record<string, any>;
  thermal: { defaultState: string; scaleToZeroAfterMinutes: number; warmupTimeSeconds: numbe
  pricing: { hourlyRate: number; perRequest: number; markup: number };
  parameters: number;
  minTier: number;
  requiresGpu: boolean;
  status: string;
  currentThermalState?: string;
  licenses: Array<{ id: string; type: string; spdx: string; commercialUse: boolean; complian
  dependencies: Array<{ name: string; type: string; sizeGb?: number; required: boolean }>;
}

export class OrchestrationEngine extends EventEmitter {
  private pool: Pool;
  private modelCache: Map<string, { config: ModelConfig; timestamp: number }> = new Map();
```

```
private cacheExpiry = 5 * 60 * 1000; // 5 minutes

constructor(pool: Pool) {
  super();
  this.pool = pool;
}

/**
 * Get model config from database - REPLACES hardcoded TypeScript configs
 */
async getModelConfig(modelId: string): Promise<ModelConfig | null> {
  const cached = this.modelCache.get(modelId);
  if (cached && Date.now() - cached.timestamp < this.cacheExpiry) {
    return cached.config;
  }

  const result = await this.pool.query('SELECT get_model_config($1) as config', [modelId])
  if (result.rows[0]?.config) {
    const config = result.rows[0].config as ModelConfig;
    this.modelCache.set(modelId, { config, timestamp: Date.now() });
    return config;
  }
  return null;
}

/**
 * List models with filters - Used by Admin Dashboard
 */
async listModels(filters?: { category?: string; status?: string; minTier?: number; provide
  let query = `SELECT get_model_config(model_id) as config FROM ai_models WHERE enabled =
  const params: any[] = [];
  let idx = 1;

  if (filters?.category) { query += ` AND category = $${idx++}`; params.push(filters.categ
  if (filters?.status) { query += ` AND status = $${idx++}`; params.push(filters.status);
  if (filters?.minTier) { query += ` AND min_tier <= $${idx++}`; params.push(filters.minTi
  if (filters?.providerType) { query += ` AND provider_type = $${idx++}`; params.push(filt

  query += ` ORDER BY category, display_name`;
  const result = await this.pool.query(query, params);
  return result.rows.map(r => r.config).filter(Boolean);
}

/**
 * Create model via Admin Dashboard - NO code deployment needed
 */
```

```
async createModel(input: any, adminId: string): Promise<string> {
  const client = await this.pool.connect();
  try {
    await client.query('BEGIN');

    const result = await client.query(`
      INSERT INTO ai_models (
        model_id, name, display_name, description, category, specialty,
        provider_type, deployment_config, parameters, accuracy, benchmark,
        capabilities, input_formats, output_formats, thermal_config, pricing_config,
        min_tier, requires_gpu, gpu_memory_gb, status, version, repository, created_by
      ) VALUES ($1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,$16,$17,$18,$19,$20,$21
      RETURNING id
    `, [
      input.modelId, input.name, input.displayName, input.description,
      input.category, input.specialty, input.providerType || 'self_hosted',
      JSON.stringify(input.deployment || {}), input.parameters || 0,
      input.accuracy, input.benchmark, input.capabilities || [],
      input.inputFormats || [], input.outputFormats || [],
      JSON.stringify(input.thermal || {}), JSON.stringify(input.pricing || {}),
      input.minTier || 1, input.requiresGpu || false, input.gpuMemoryGb || 0,
      input.status || 'active', input.version, input.repository, adminId
    ]);

    const modelUuid = result.rows[0].id;

    // Insert licenses
    for (const license of (input.licenses || [])) {
      await client.query(
        `INSERT INTO model_licenses (model_id, license_type, license_spdx, license_url, co
        [modelUuid, license.type, license.spdx, license.url, license.commercialUse ?? true
      );
    }

    // Insert dependencies
    for (const dep of (input.dependencies || [])) {
      await client.query(
        `INSERT INTO model_dependencies (model_id, dependency_type, dependency_name, depen
        [modelUuid, dep.type, dep.name, dep.sizeGb, dep.license, dep.required ?? true]
      );
    }

    await client.query('COMMIT');
    this.modelCache.delete(input.modelId);
    this.emit('model:created', { modelId: input.modelId, adminId });
    return modelUuid;
```

```
  } catch (error) {
    await client.query('ROLLBACK');
    throw error;
  } finally {
    client.release();
  }
}

/**
 * Update model via Admin Dashboard
 */
async updateModel(modelId: string, updates: any, adminId: string): Promise<void> {
  const sets: string[] = [];
  const params: any[] = [];
  let idx = 1;

  const fields: Record<string, string> = {
    displayName: 'display_name', description: 'description', category: 'category',
    status: 'status', minTier: 'min_tier', version: 'version'
  };

  for (const [key, col] of Object.entries(fields)) {
    if (updates[key] !== undefined) { sets.push(`${col} = $${idx++}`); params.push(updates
  }

  if (updates.deployment) { sets.push(`deployment_config = $${idx++}`); params.push(JSON.s
  if (updates.thermal) { sets.push(`thermal_config = $${idx++}`); params.push(JSON.stringi
  if (updates.pricing) { sets.push(`pricing_config = $${idx++}`); params.push(JSON.stringi

  if (sets.length === 0) return;

  sets.push(`updated_at = NOW()`, `updated_by = $${idx++}`);
  params.push(adminId, modelId);

  await this.pool.query(`UPDATE ai_models SET ${sets.join(', ')} WHERE model_id = $${idx}`
  this.modelCache.delete(modelId);
  this.emit('model:updated', { modelId, adminId });
}

/**
 * Delete model (soft delete)
 */
async deleteModel(modelId: string, adminId: string): Promise<void> {
  await this.pool.query(
    `UPDATE ai_models SET enabled = false, status = 'disabled', updated_at = NOW(), update
    [modelId, adminId]
```

```
  );
  this.modelCache.delete(modelId);
  this.emit('model:deleted', { modelId, adminId });
}

/**
 * Get workflow config from database
 */
async getWorkflowConfig(workflowId: string): Promise<any> {
  const result = await this.pool.query('SELECT get_workflow_config($1) as config', [workfl
  return result.rows[0]?.config || null;
}

/**
 * Execute workflow with parameters
 */
async executeWorkflow(workflowId: string, tenantId: string, userId: string, parameters: Re
  const workflow = await this.getWorkflowConfig(workflowId);
  if (!workflow) throw new Error(`Workflow not found: ${workflowId}`);

  const resolvedParams = { ...workflow.defaultParameters, ...parameters };

  const result = await this.pool.query(`
    INSERT INTO workflow_executions (workflow_id, tenant_id, user_id, status, input_parame
    VALUES ((SELECT id FROM workflow_definitions WHERE workflow_id = $1), $2, $3, 'running
    RETURNING id
  `, [workflowId, tenantId, userId, JSON.stringify(parameters), JSON.stringify(resolvedPar

  const executionId = result.rows[0].id;
  this.emit('workflow:started', { executionId, workflowId, tenantId, userId });
  return executionId;
}

/**
 * Get license summary for compliance dashboard
 */
async getLicenseSummary(): Promise<{ totalModels: number; commercialOk: number; reviewNeed
  const result = await this.pool.query(`
    SELECT
      COUNT(DISTINCT m.id) as total_models,
      COUNT(DISTINCT CASE WHEN NOT EXISTS (SELECT 1 FROM model_licenses l2 WHERE l2.model_
      COUNT(DISTINCT CASE WHEN l.compliance_status = 'review_needed' THEN m.id END) as rev
      COUNT(DISTINCT CASE WHEN l.compliance_status = 'non_compliant' THEN m.id END) as non
      COUNT(DISTINCT CASE WHEN l.expires_at IS NOT NULL AND l.expires_at < NOW() + INTERVA
    FROM ai_models m LEFT JOIN model_licenses l ON m.id = l.model_id WHERE m.enabled = tru
  `);
```

```
    return {
      totalModels: parseInt(result.rows[0].total_models),
      commercialOk: parseInt(result.rows[0].commercial_ok),
      reviewNeeded: parseInt(result.rows[0].review_needed),
      nonCompliant: parseInt(result.rows[0].non_compliant),
      expiringIn30Days: parseInt(result.rows[0].expiring_soon),
    };
  }

  refreshCache(): void {
    this.modelCache.clear();
  }
}

let instance: OrchestrationEngine | null = null;
export function getOrchestrationEngine(pool: Pool): OrchestrationEngine {
  if (!instance) instance = new OrchestrationEngine(pool);
  return instance;
}
```

SECTION-35-LICENSE-MANAGEMENT

SECTION 35: ADMIN DASHBOARD - MODEL & LICENSE MANAGEMENT UI (v4.1.0)

Full CRUD UI for managing AI models through the Admin Dashboard. Administrators can add/edit/delete models without any code changes.

35.1 MODEL MANAGEMENT PAGE

apps/admin-dashboard/app/(dashboard)/models/page.tsx

```
'use client';

import { useState } from 'react';
```

```tsx
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { Plus, Search, RefreshCw, AlertTriangle } from 'lucide-react';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs';
import { ModelTable } from '@/components/models/model-table';
import { ModelForm } from '@/components/models/model-form';
import { LicensePanel } from '@/components/models/license-panel';
import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogTrigger } from '@/componen
import { toast } from '@/components/ui/use-toast';
import { api } from '@/lib/api';

export default function ModelsPage() {
  const [searchQuery, setSearchQuery] = useState('');
  const [categoryFilter, setCategoryFilter] = useState('all');
  const [showAddModel, setShowAddModel] = useState(false);
  const [editingModel, setEditingModel] = useState<string | null>(null);
  const queryClient = useQueryClient();

  const { data: models, isLoading, refetch } = useQuery({
    queryKey: ['models', categoryFilter],
    queryFn: async () => {
      const params = categoryFilter !== 'all' ? `?category=${categoryFilter}` : '';
      const response = await api.get(`/api/v2/admin/models${params}`);
      return response.data;
    },
  });

  const { data: licenseSummary } = useQuery({
    queryKey: ['license-summary'],
    queryFn: async () => (await api.get('/api/v2/admin/models/licenses/summary')).data,
  });

  const createModelMutation = useMutation({
    mutationFn: (data: any) => api.post('/api/v2/admin/models', data),
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['models'] });
      setShowAddModel(false);
      toast({ title: 'Model Created', description: 'New model added to registry.' });
    },
  });

  const deleteModelMutation = useMutation({
    mutationFn: (modelId: string) => api.delete(`/api/v2/admin/models/${modelId}`),
    onSuccess: () => {
```

```
    queryClient.invalidateQueries({ queryKey: ['models'] });
    toast({ title: 'Model Deleted' });
  },
});

const filteredModels = models?.filter((m: any) =>
  m.displayName.toLowerCase().includes(searchQuery.toLowerCase()) ||
  m.modelId.toLowerCase().includes(searchQuery.toLowerCase())
) || [];

return (
  <div className="space-y-6">
    <div className="flex items-center justify-between">
      <div>
        <h1 className="text-3xl font-bold">AI Models</h1>
        <p className="text-muted-foreground">
          Database-driven model management - no code changes needed!
        </p>
      </div>
      <div className="flex gap-2">
        <Button variant="outline" size="sm" onClick={() => refetch()}>
          <RefreshCw className="h-4 w-4 mr-2" /> Refresh
        </Button>
        <Dialog open={showAddModel} onOpenChange={setShowAddModel}>
          <DialogTrigger asChild>
            <Button><Plus className="h-4 w-4 mr-2" /> Add Model</Button>
          </DialogTrigger>
          <DialogContent className="max-w-4xl max-h-[90vh] overflow-y-auto">
            <DialogHeader>
              <DialogTitle>Add New AI Model</DialogTitle>
            </DialogHeader>
            <ModelForm
              onSubmit={(data) => createModelMutation.mutate(data)}
              isLoading={createModelMutation.isPending}
              onCancel={() => setShowAddModel(false)}
            />
          </DialogContent>
        </Dialog>
      </div>
    </div>

    {/* Summary Cards */}
    <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
      <Card>
        <CardHeader className="pb-2"><CardTitle className="text-sm text-muted-foreground">
        <CardContent><div className="text-2xl font-bold">{licenseSummary?.totalModels || (
```

```
  </Card>
  <Card>
    <CardHeader className="pb-2"><CardTitle className="text-sm text-muted-foreground">
    <CardContent><div className="text-2xl font-bold text-green-600">{licenseSummary?.c
  </Card>
  <Card>
    <CardHeader className="pb-2"><CardTitle className="text-sm text-muted-foreground">
    <CardContent><div className="text-2xl font-bold text-yellow-600">{licenseSummary?.
  </Card>
  <Card>
    <CardHeader className="pb-2"><CardTitle className="text-sm text-muted-foreground">
    <CardContent><div className="text-2xl font-bold text-orange-600">{licenseSummary?.
  </Card>
</div>

{(licenseSummary?.nonCompliant || 0) > 0 && (
  <Card className="border-red-200 bg-red-50">
    <CardContent className="flex items-center gap-4 py-4">
      <AlertTriangle className="h-8 w-8 text-red-600" />
      <div>
        <h3 className="font-semibold text-red-800">License Compliance Alert</h3>
        <p className="text-red-700">{licenseSummary?.nonCompliant} model(s) have non-c
      </div>
    </CardContent>
  </Card>
)}

<Tabs defaultValue="models" className="space-y-4">
  <TabsList>
    <TabsTrigger value="models">Models</TabsTrigger>
    <TabsTrigger value="licenses">Licenses</TabsTrigger>
    <TabsTrigger value="workflows">Workflows</TabsTrigger>
  </TabsList>

  <TabsContent value="models" className="space-y-4">
    <div className="flex gap-4">
      <div className="relative flex-1">
        <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-muted
        <Input
          placeholder="Search models..."
          value={searchQuery}
          onChange={(e) => setSearchQuery(e.target.value)}
          className="pl-10"
        />
      </div>
      <select
```

```
              value={categoryFilter}
              onChange={(e) => setCategoryFilter(e.target.value)}
              className="px-3 py-2 border rounded-md"
            >
              <option value="all">All Categories</option>
              <option value="scientific_protein">  Protein Folding</option>
              <option value="vision_detection">  Object Detection</option>
              <option value="audio_stt">  Speech-to-Text</option>
              <option value="medical_imaging">  Medical Imaging</option>
              <option value="llm">  LLM</option>
            </select>
          </div>

          <ModelTable
            models={filteredModels}
            isLoading={isLoading}
            onEdit={setEditingModel}
            onDelete={(id) => confirm('Delete model?') && deleteModelMutation.mutate(id)}
          />
        </TabsContent>

        <TabsContent value="licenses"><LicensePanel /></TabsContent>
        <TabsContent value="workflows"><p className="text-muted-foreground">Workflow manager
      </Tabs>
    </div>
  );
}
```

35.2 API ENDPOINTS

packages/lambda/admin/orchestration.ts

```
/**
 * RADIANT v4.1.0 - Admin Orchestration API
 */

import { APIGatewayProxyHandler } from 'aws-lambda';
import { Pool } from 'pg';
import { getOrchestrationEngine } from '@radiant/services/orchestration';
import { requirePermission } from '../auth/permissions';
import { createResponse, createErrorResponse } from '../utils/response';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });

export const listModels: APIGatewayProxyHandler = async (event) => {
```

```
  try {
    await requirePermission(event, 'models:read');
    const { category, status, minTier, providerType } = event.queryStringParameters || {};
    const engine = getOrchestrationEngine(pool);
    const models = await engine.listModels({ category, status, minTier: minTier ? parseInt(m
    return createResponse(200, models);
  } catch (error: any) {
    return createErrorResponse(error);
  }
};

export const getModel: APIGatewayProxyHandler = async (event) => {
  try {
    await requirePermission(event, 'models:read');
    const modelId = event.pathParameters?.modelId;
    if (!modelId) return createResponse(400, { error: 'Model ID required' });

    const engine = getOrchestrationEngine(pool);
    const model = await engine.getModelConfig(modelId);
    if (!model) return createResponse(404, { error: 'Model not found' });
    return createResponse(200, model);
  } catch (error: any) {
    return createErrorResponse(error);
  }
};

export const createModel: APIGatewayProxyHandler = async (event) => {
  try {
    const admin = await requirePermission(event, 'models:write');
    const data = JSON.parse(event.body || '{}');
    const engine = getOrchestrationEngine(pool);
    const modelUuid = await engine.createModel(data, admin.id);
    return createResponse(201, { id: modelUuid, modelId: data.modelId, message: 'Model creat
  } catch (error: any) {
    return createErrorResponse(error);
  }
};

export const updateModel: APIGatewayProxyHandler = async (event) => {
  try {
    const admin = await requirePermission(event, 'models:write');
    const modelId = event.pathParameters?.modelId;
    if (!modelId) return createResponse(400, { error: 'Model ID required' });

    const updates = JSON.parse(event.body || '{}');
    const engine = getOrchestrationEngine(pool);
```

```
    await engine.updateModel(modelId, updates, admin.id);
    return createResponse(200, { success: true });
  } catch (error: any) {
    return createErrorResponse(error);
  }
};

export const deleteModel: APIGatewayProxyHandler = async (event) => {
  try {
    const admin = await requirePermission(event, 'models:write');
    const modelId = event.pathParameters?.modelId;
    if (!modelId) return createResponse(400, { error: 'Model ID required' });

    const engine = getOrchestrationEngine(pool);
    await engine.deleteModel(modelId, admin.id);
    return createResponse(200, { success: true });
  } catch (error: any) {
    return createErrorResponse(error);
  }
};

export const getLicenseSummary: APIGatewayProxyHandler = async (event) => {
  try {
    await requirePermission(event, 'models:read');
    const engine = getOrchestrationEngine(pool);
    const summary = await engine.getLicenseSummary();
    return createResponse(200, summary);
  } catch (error: any) {
    return createErrorResponse(error);
  }
};

export const listWorkflows: APIGatewayProxyHandler = async (event) => {
  try {
    await requirePermission(event, 'workflows:read');
    const result = await pool.query(`
      SELECT workflow_id, name, description, category, version, min_tier
      FROM workflow_definitions WHERE enabled = true ORDER BY category, name
    `);
    return createResponse(200, result.rows);
  } catch (error: any) {
    return createErrorResponse(error);
  }
};

export const executeWorkflow: APIGatewayProxyHandler = async (event) => {
```

```
  try {
    const admin = await requirePermission(event, 'workflows:execute');
    const workflowId = event.pathParameters?.workflowId;
    if (!workflowId) return createResponse(400, { error: 'Workflow ID required' });

    const { tenantId, userId, parameters } = JSON.parse(event.body || '{}');
    const engine = getOrchestrationEngine(pool);
    const executionId = await engine.executeWorkflow(workflowId, tenantId, userId, parameter
    return createResponse(202, { executionId, status: 'started' });
  } catch (error: any) {
    return createErrorResponse(error);
  }
};
```

35.3 VERIFICATION COMMANDS

```
# Apply orchestration migration
psql $DATABASE_URL -f packages/database/migrations/034_orchestration_engine.sql

# Apply AlphaFold 2 seed data
psql $DATABASE_URL -f packages/database/migrations/034a_seed_alphafold2.sql

# Verify AlphaFold 2 is in registry
psql $DATABASE_URL -c "SELECT model_id, display_name, status FROM ai_models WHERE model_id =

# Verify licenses
psql $DATABASE_URL -c "SELECT m.display_name, l.license_spdx, l.commercial_use FROM model_li

# Test get_model_config function
psql $DATABASE_URL -c "SELECT get_model_config('alphafold2')" | head -50

# Test get_workflow_config function
psql $DATABASE_URL -c "SELECT get_workflow_config('protein_folding_alphafold2')" | head -50

# Verify audit log is recording
psql $DATABASE_URL -c "SELECT entity_type, action, change_summary, created_at FROM orchestra
```

SECTION-36-UNIFIED-MODEL-REGISTRY

SECTION 36: UNIFIED MODEL REGISTRY & SYNC SERVICE (v4.2.0)

Section 36 of 37 | Depends on: Sections 0-35 | Creates: Unified registry, sync service, complete model catalog

36.1 OVERVIEW

This section creates: 1. **Unified Model Registry** - SQL view combining ALL 106 models (50+ external + 56 self-hosted) 2. **Registry Sync Service** - Automated Lambda for provider/model synchronization 3. **Complete Self-Hosted Model Catalog** - 56 models with full metadata 4. **Orchestration Model Selection** - Smart selection algorithm with thermal awareness 5. **Health Monitoring** - Provider/endpoint health tracking

---

36.2 DATABASE SCHEMA

packages/database/migrations/036_unified_model_registry.sql

```
-- ============================================================================
-- RADIANT v4.2.0 - Unified Model Registry Migration
-- ============================================================================
-- Combines external providers (21) and self-hosted models (56+) into single view
-- Provides orchestration engine with complete model selection metadata
-- ============================================================================

-- ============================================================================
-- SELF-HOSTED MODELS CATALOG (56 models)
-- ============================================================================

CREATE TABLE IF NOT EXISTS self_hosted_models (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    model_id VARCHAR(100) NOT NULL UNIQUE,
    name VARCHAR(255) NOT NULL,
    display_name VARCHAR(255) NOT NULL,
    description TEXT,

    -- Categorization
    category VARCHAR(50) NOT NULL,  -- vision, audio, scientific, medical, geospatial, 3d, 
    specialty VARCHAR(50) NOT NULL, -- object_detection, protein_folding, etc.
```

```
-- Capabilities & Modalities
capabilities TEXT[] NOT NULL DEFAULT '{}',
input_modalities TEXT[] NOT NULL DEFAULT '{}'::TEXT[],
output_modalities TEXT[] NOT NULL DEFAULT '{}'::TEXT[],
primary_mode VARCHAR(20) NOT NULL DEFAULT 'inference',

-- SageMaker Configuration
sagemaker_image VARCHAR(500) NOT NULL,
instance_type VARCHAR(50) NOT NULL,
gpu_memory_gb INTEGER NOT NULL,
environment JSONB NOT NULL DEFAULT '{}',
model_data_url TEXT,

-- Model Specs
parameters BIGINT,
accuracy VARCHAR(100),
benchmark VARCHAR(255),
context_window INTEGER,
max_output INTEGER,

-- I/O Formats
input_formats TEXT[] NOT NULL DEFAULT '{}',
output_formats TEXT[] NOT NULL DEFAULT '{}',

-- Licensing
license VARCHAR(100) NOT NULL,
license_url TEXT,
commercial_use_allowed BOOLEAN NOT NULL DEFAULT true,
commercial_use_notes TEXT,
attribution_required BOOLEAN NOT NULL DEFAULT false,

-- Pricing (75% markup on SageMaker costs)
hourly_rate DECIMAL(10,4) NOT NULL,
per_request DECIMAL(10,6),
per_image DECIMAL(10,6),
per_minute_audio DECIMAL(10,6),
per_minute_video DECIMAL(10,6),
per_3d_model DECIMAL(10,4),
markup_percent DECIMAL(5,2) NOT NULL DEFAULT 75.00,

-- Tier Requirements
min_tier INTEGER NOT NULL DEFAULT 3,  -- Self-hosted requires Tier 3+

-- Thermal Defaults
default_thermal_state VARCHAR(20) NOT NULL DEFAULT 'COLD',
```

```
    warmup_time_seconds INTEGER NOT NULL DEFAULT 60,
    scale_to_zero_minutes INTEGER NOT NULL DEFAULT 15,
    min_instances INTEGER NOT NULL DEFAULT 0,
    max_instances INTEGER NOT NULL DEFAULT 3,

    -- Status
    status VARCHAR(20) NOT NULL DEFAULT 'active',
    enabled BOOLEAN NOT NULL DEFAULT true,
    deprecated BOOLEAN NOT NULL DEFAULT false,

    -- Metadata
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_self_hosted_category ON self_hosted_models(category);
CREATE INDEX idx_self_hosted_specialty ON self_hosted_models(specialty);
CREATE INDEX idx_self_hosted_status ON self_hosted_models(status);
CREATE INDEX idx_self_hosted_enabled ON self_hosted_models(enabled);

-- ============================================================================
-- PROVIDER HEALTH MONITORING
-- ============================================================================

CREATE TABLE IF NOT EXISTS provider_health (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    provider_id VARCHAR(50) NOT NULL REFERENCES providers(id),
    region VARCHAR(50) NOT NULL DEFAULT 'us-east-1',

    -- Health Status
    status VARCHAR(20) NOT NULL DEFAULT 'unknown', -- healthy, degraded, unhealthy, unknown
    avg_latency_ms INTEGER,
    p95_latency_ms INTEGER,
    p99_latency_ms INTEGER,
    error_rate DECIMAL(5, 2),
    success_rate DECIMAL(5, 2),

    -- Last Check
    last_check_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    last_success_at TIMESTAMPTZ,
    last_failure_at TIMESTAMPTZ,
    last_error TEXT,

    -- Metadata
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
```

```
    UNIQUE(provider_id, region)
);

CREATE INDEX idx_provider_health_provider ON provider_health(provider_id);
CREATE INDEX idx_provider_health_status ON provider_health(status);

-- ============================================================================
-- REGISTRY SYNC LOG
-- ============================================================================

CREATE TABLE IF NOT EXISTS registry_sync_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    sync_type VARCHAR(50) NOT NULL, -- full, health, pricing, models

    -- Results
    providers_updated INTEGER NOT NULL DEFAULT 0,
    models_added INTEGER NOT NULL DEFAULT 0,
    models_updated INTEGER NOT NULL DEFAULT 0,
    models_deprecated INTEGER NOT NULL DEFAULT 0,
    errors TEXT[],

    -- Timing
    started_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    completed_at TIMESTAMPTZ,
    duration_ms INTEGER,

    -- Status
    status VARCHAR(20) NOT NULL DEFAULT 'running', -- running, completed, failed
    error_message TEXT
);

CREATE INDEX idx_registry_sync_type ON registry_sync_log(sync_type);
CREATE INDEX idx_registry_sync_status ON registry_sync_log(status);
CREATE INDEX idx_registry_sync_started ON registry_sync_log(started_at DESC);

-- ============================================================================
-- UNIFIED MODEL REGISTRY VIEW
-- ============================================================================

CREATE OR REPLACE VIEW unified_model_registry AS
-- External Provider Models
SELECT
    m.id::TEXT AS id,
    m.provider_id,
    p.display_name AS provider_name,
```

```
m.model_id,
m.litellm_id,
m.name,
m.display_name,
m.description,

-- Hosting Type
'external' AS hosting_type,

-- Category & Modality
m.category,
m.capabilities,
m.input_modalities,
m.output_modalities,

-- Primary Mode (derived)
CASE
    WHEN 'chat' = ANY(m.capabilities) THEN 'chat'
    WHEN 'completion' = ANY(m.capabilities) THEN 'completion'
    WHEN 'embedding' = ANY(m.capabilities) OR m.category = 'embedding' THEN 'embedding'
    WHEN m.category = 'image_generation' THEN 'image'
    WHEN m.category = 'video_generation' THEN 'video'
    WHEN m.category IN ('audio_generation', 'text_to_speech') THEN 'audio'
    WHEN m.category = 'speech_to_text' THEN 'transcription'
    WHEN m.category = 'search' THEN 'search'
    WHEN m.category = '3d_generation' THEN '3d'
    ELSE 'other'
END AS primary_mode,

-- Context & Limits
m.context_window,
m.max_output,

-- Pricing
m.pricing_type,
m.input_cost_per_1k,
m.output_cost_per_1k,
m.cost_per_request,
m.cost_per_second,
m.cost_per_image,
m.cost_per_minute,
m.markup_rate,

-- Self-Hosted Specific (NULL for external)
NULL::VARCHAR AS instance_type,
NULL::INTEGER AS gpu_memory_gb,
```

```
    NULL::VARCHAR AS thermal_state,
    NULL::BOOLEAN AS is_transitioning,
    NULL::INTEGER AS warmup_time_seconds,

    -- Status
    m.enabled,
    m.deprecated,
    ph.status AS health_status,
    ph.avg_latency_ms,
    ph.error_rate,

    -- Compliance
    p.compliance,
    NULL::VARCHAR AS license,
    TRUE AS commercial_use_allowed,

    -- Tier
    1 AS min_tier,  -- External available to all tiers

    -- Timestamps
    m.created_at,
    m.updated_at

FROM models m
JOIN providers p ON m.provider_id = p.id
LEFT JOIN provider_health ph ON p.id = ph.provider_id AND ph.region = 'us-east-1'
WHERE m.enabled = true AND p.enabled = true

UNION ALL

-- Self-Hosted Models
SELECT
    sh.id::TEXT AS id,
    'self_hosted' AS provider_id,
    'RADIANT Self-Hosted' AS provider_name,
    sh.model_id,
    'sagemaker/' || sh.model_id AS litellm_id,
    sh.name,
    sh.display_name,
    sh.description,

    -- Hosting Type
    'self_hosted' AS hosting_type,

    -- Category & Modality
    sh.category,
```

```
    sh.capabilities,
    sh.input_modalities,
    sh.output_modalities,
    sh.primary_mode,

    -- Context & Limits
    sh.context_window,
    sh.max_output,

    -- Pricing
    'per_hour'::VARCHAR AS pricing_type,
    NULL::NUMERIC AS input_cost_per_1k,
    NULL::NUMERIC AS output_cost_per_1k,
    sh.per_request AS cost_per_request,
    NULL::NUMERIC AS cost_per_second,
    sh.per_image AS cost_per_image,
    sh.per_minute_audio AS cost_per_minute,
    sh.markup_percent / 100 AS markup_rate,

    -- Self-Hosted Specific
    sh.instance_type,
    sh.gpu_memory_gb,
    ts.current_state AS thermal_state,
    ts.is_transitioning,
    sh.warmup_time_seconds,

    -- Status
    sh.enabled,
    sh.deprecated,
    CASE WHEN ts.current_state IN ('WARM', 'HOT') THEN 'healthy' ELSE 'unknown' END AS healt
    NULL::INTEGER AS avg_latency_ms,
    NULL::NUMERIC AS error_rate,

    -- Compliance
    ARRAY[]::TEXT[] AS compliance,
    sh.license,
    sh.commercial_use_allowed,

    -- Tier
    sh.min_tier,

    -- Timestamps
    sh.created_at,
    sh.updated_at

FROM self_hosted_models sh
```

```sql
LEFT JOIN thermal_states ts ON sh.model_id = ts.model_id
WHERE sh.enabled = true;

-- Index on the view (for performance)
CREATE INDEX IF NOT EXISTS idx_models_hosting_type ON models((CASE WHEN is_self_hosted THEN

-- ============================================================================
-- MODEL SELECTION FUNCTION
-- ============================================================================

CREATE OR REPLACE FUNCTION select_model(
    p_task VARCHAR(20),
    p_input_modalities TEXT[],
    p_output_modalities TEXT[],
    p_tenant_tier INTEGER,
    p_prefer_hosting VARCHAR(20) DEFAULT 'any',
    p_required_capabilities TEXT[] DEFAULT '{}'::TEXT[],
    p_min_context_window INTEGER DEFAULT NULL,
    p_require_hipaa BOOLEAN DEFAULT FALSE
)
RETURNS TABLE (
    model_id VARCHAR,
    display_name VARCHAR,
    hosting_type VARCHAR,
    provider_name VARCHAR,
    primary_mode VARCHAR,
    thermal_state VARCHAR,
    warmup_required BOOLEAN,
    warmup_time_seconds INTEGER,
    health_status VARCHAR
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        u.model_id,
        u.display_name,
        u.hosting_type,
        u.provider_name,
        u.primary_mode,
        u.thermal_state,
        (u.hosting_type = 'self_hosted' AND u.thermal_state = 'COLD') AS warmup_required,
        u.warmup_time_seconds,
        u.health_status
    FROM unified_model_registry u
    WHERE
        -- Task/mode match
```

```
        u.primary_mode = p_task
        -- Modality match
        AND p_input_modalities <@ u.input_modalities
        AND p_output_modalities <@ u.output_modalities
        -- Tier eligibility
        AND u.min_tier <= p_tenant_tier
        -- Not unhealthy
        AND (u.health_status IS NULL OR u.health_status != 'unhealthy')
        -- Hosting preference
        AND (p_prefer_hosting = 'any' OR u.hosting_type = p_prefer_hosting)
        -- Required capabilities
        AND (p_required_capabilities = '{}'::TEXT[] OR p_required_capabilities <@ u.capabili
        -- Context window
        AND (p_min_context_window IS NULL OR u.context_window >= p_min_context_window)
        -- HIPAA compliance
        AND (NOT p_require_hipaa OR 'HIPAA' = ANY(u.compliance))
    ORDER BY
        -- Prefer HOT > WARM > COLD for latency
        CASE u.thermal_state
            WHEN 'HOT' THEN 0
            WHEN 'WARM' THEN 1
            WHEN 'COLD' THEN 2
            ELSE 3
        END,
        -- Then by latency
        u.avg_latency_ms ASC NULLS LAST,
        -- Then by health
        CASE u.health_status
            WHEN 'healthy' THEN 0
            WHEN 'degraded' THEN 1
            ELSE 2
        END
    LIMIT 10;
END;
$$ LANGUAGE plpgsql;

-- ============================================================================
-- TRIGGERS FOR UPDATED_AT
-- ============================================================================

CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_self_hosted_models_updated_at
    BEFORE UPDATE ON self_hosted_models
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

CREATE TRIGGER update_provider_health_updated_at
    BEFORE UPDATE ON provider_health
    FOR EACH ROW EXECUTE FUNCTION update_updated_at_column();

-- ============================================================================
-- INITIAL DATA INSERT
-- ============================================================================

INSERT INTO schema_migrations (version, name, applied_by)
VALUES ('036', 'unified_model_registry', 'system')
ON CONFLICT (version) DO NOTHING;
```

36.3 SELF-HOSTED MODEL SEED DATA

packages/database/migrations/036a_seed_self_hosted_models.sql

```
-- ============================================================================
-- RADIANT v4.2.0 - Self-Hosted Models Seed Data (56 Models)
-- ============================================================================

-- ============================================================================
-- COMPUTER VISION MODELS (13 models)
-- ============================================================================

INSERT INTO self_hosted_models (model_id, name, display_name, description, category, special

-- Classification (4)
('efficientnet-b0', 'efficientnet-b0', 'EfficientNet-B0', 'Lightweight image classification
('efficientnetv2-l', 'efficientnetv2-l', 'EfficientNetV2-L', 'State-of-the-art classificatio
('convnext-xl', 'convnext-xl', 'ConvNeXt-XL', 'Pure ConvNet achieving transformer-level perf
('vit-l-14', 'vit-l-14', 'ViT-L/14', 'Vision Transformer Large with 14x14 patches', 'vision'

-- Detection (4)
('yolov8m', 'yolov8m', 'YOLOv8m', 'Medium YOLOv8 for real-time object detection', 'vision',
('yolov8x', 'yolov8x', 'YOLOv8x', 'Extra-large YOLOv8 for maximum accuracy', 'vision', 'dete
('yolo11m', 'yolo11m', 'YOLO11m', 'Latest YOLO generation with improved architecture', 'visi
('detr-resnet-101', 'detr-resnet-101', 'DETR-ResNet-101', 'End-to-end transformer detector',

-- Segmentation (2)
```

```
('sam-vit-h', 'sam-vit-h', 'SAM-ViT-H', 'Segment Anything Model - ViT-Huge backbone', 'visio
('sam-2', 'sam-2', 'SAM 2', 'Segment Anything Model 2 - video and image segmentation', 'visi

-- Embedding (1)
('clip-vit-l', 'clip-vit-l', 'CLIP-ViT-L', 'Contrastive Language-Image Pre-training', 'visio

-- OCR (2)
('paddleocr-v4', 'paddleocr-v4', 'PaddleOCR-v4', 'Multi-language OCR with detection and reco
('trocr-large', 'trocr-large', 'TrOCR-Large', 'Transformer-based OCR for handwritten text',

-- ============================================================================
-- AUDIO/SPEECH MODELS (6 models)
-- ============================================================================

INSERT INTO self_hosted_models (model_id, name, display_name, description, category, special

('whisper-large-v3', 'whisper-large-v3', 'Whisper-Large-v3', 'OpenAI multilingual speech rec
('whisper-large-v3-turbo', 'whisper-large-v3-turbo', 'Whisper-Large-v3-Turbo', 'Faster Whisp
('wav2vec2-xlsr-53', 'wav2vec2-xlsr-53', 'Wav2Vec2-XLSR-53', 'Cross-lingual speech represent
('titanet-l', 'titanet-l', 'TitaNet-L', 'NVIDIA speaker embedding and verification', 'audio'
('pyannote-diarization-3.1', 'pyannote-diarization-3.1', 'pyannote Speaker Diarization 3.1',
('speecht5-tts', 'speecht5-tts', 'SpeechT5 TTS', 'Microsoft text-to-speech synthesis', 'audi

-- ============================================================================
-- SCIENTIFIC COMPUTING MODELS (8 models)
-- ============================================================================

INSERT INTO self_hosted_models (model_id, name, display_name, description, category, special

('alphafold2', 'alphafold2', 'AlphaFold 2', 'Nobel Prize-winning protein structure predictio
('esm2-650m', 'esm2-650m', 'ESM-2 (650M)', 'Meta protein language model - medium', 'scientif
('esm2-3b', 'esm2-3b', 'ESM-2 (3B)', 'Meta protein language model - large', 'scientific', 'p
('esmfold', 'esmfold', 'ESMFold', 'Single-sequence protein structure prediction', 'scientifi
('rosettafold2', 'rosettafold2', 'RoseTTAFold2', 'Protein complex structure prediction', 'sc
('alphageometry', 'alphageometry', 'AlphaGeometry', 'Olympiad-level geometry reasoning', 'sc
('muzero', 'muzero', 'MuZero', 'DeepMind model-based planning', 'scientific', 'planning', AP
('graphormer', 'graphormer', 'Graphormer', 'Transformer for molecular property prediction',

-- ============================================================================
-- MEDICAL IMAGING MODELS (6 models)
-- ============================================================================

INSERT INTO self_hosted_models (model_id, name, display_name, description, category, special

('nnunet', 'nnunet', 'nnU-Net', 'Self-configuring medical image segmentation', 'medical', 's
('medsam', 'medsam', 'MedSAM', 'Segment Anything for Medical Images', 'medical', 'segmentati
```

```
('med-sam2', 'med-sam2', 'Med-SAM2', 'Medical SAM 2 for 3D and video', 'medical', 'segmentat
('biomedclip', 'biomedclip', 'BiomedCLIP', 'Medical image-text embeddings', 'medical', 'embe
('chexnet', 'chexnet', 'CheXNet', 'Chest X-ray pathology detection', 'medical', 'classificat
('monai-vista3d', 'monai-vista3d', 'MONAI VISTA-3D', '3D medical image segmentation foundati

-- ============================================================================
-- GEOSPATIAL MODELS (4 models)
-- ============================================================================

INSERT INTO self_hosted_models (model_id, name, display_name, description, category, special

('prithvi-100m', 'prithvi-100m', 'Prithvi-100M', 'NASA/IBM geospatial foundation model', 'ge
('prithvi-600m', 'prithvi-600m', 'Prithvi-600M', 'NASA/IBM large geospatial model', 'geospat
('satmae', 'satmae', 'SatMAE', 'Self-supervised satellite image analysis', 'geospatial', 'fo
('geosam', 'geosam', 'GeoSAM', 'Segment Anything for geospatial', 'geospatial', 'segmentatio

-- ============================================================================
-- 3D/RECONSTRUCTION MODELS (5 models)
-- ============================================================================

INSERT INTO self_hosted_models (model_id, name, display_name, description, category, special

('nerfstudio-nerfacto', 'nerfstudio-nerfacto', 'Nerfstudio Nerfacto', 'Real-time NeRF scene
('3dgs', '3dgs', '3D Gaussian Splatting', 'Real-time radiance field rendering', '3d', 'splat
('instant-ngp', 'instant-ngp', 'Instant-NGP', 'NVIDIA instant neural graphics primitives', '
('point-e', 'point-e', 'Point-E', 'OpenAI text-to-3D point cloud', '3d', 'generation', ARRAY
('shap-e', 'shap-e', 'Shap-E', 'OpenAI text/image to 3D mesh', '3d', 'generation', ARRAY['te

-- ============================================================================
-- LLM/EMBEDDINGS MODELS (14 models)
-- ============================================================================

INSERT INTO self_hosted_models (model_id, name, display_name, description, category, special

-- Large LLMs
('llama-3.3-70b', 'llama-3.3-70b', 'Llama 3.3 70B', 'Meta latest flagship LLM', 'llm', 'chat
('llama-3.2-11b-vision', 'llama-3.2-11b-vision', 'Llama 3.2 11B Vision', 'Meta multimodal LL
('mistral-7b-v0.3', 'mistral-7b-v0.3', 'Mistral 7B v0.3', 'Mistral efficient base model', 'l
('mixtral-8x7b', 'mixtral-8x7b', 'Mixtral 8x7B', 'Mistral mixture of experts', 'llm', 'chat'
('qwen2.5-72b', 'qwen2.5-72b', 'Qwen2.5 72B', 'Alibaba flagship LLM', 'llm', 'chat', ARRAY['

-- Code Models
('codellama-70b', 'codellama-70b', 'CodeLlama 70B', 'Meta code-specialized LLM', 'llm', 'cod
('starcoder2-15b', 'starcoder2-15b', 'StarCoder2 15B', 'BigCode multi-language code model',
('deepseek-coder-33b', 'deepseek-coder-33b', 'DeepSeek Coder 33B', 'DeepSeek coding speciali
```

```sql
-- Embeddings
('bge-large-en', 'bge-large-en', 'BGE-Large-EN', 'BAAI general embedding model', 'llm', 'emb
('bge-m3', 'bge-m3', 'BGE-M3', 'Multi-lingual multi-function embeddings', 'llm', 'embedding'
('e5-mistral-7b', 'e5-mistral-7b', 'E5-Mistral-7B', 'Mistral-based embeddings', 'llm', 'embe
('jina-embeddings-v3', 'jina-embeddings-v3', 'Jina Embeddings v3', 'Jina multi-task embeddin
('mxbai-embed-large', 'mxbai-embed-large', 'mxbai-embed-large', 'Mixedbread high-quality emb
('gte-qwen2-7b', 'gte-qwen2-7b', 'GTE-Qwen2-7B', 'Alibaba instruction-tuned embeddings', 'll

-- Update schema migrations
INSERT INTO schema_migrations (version, name, applied_by)
VALUES ('036a', 'seed_self_hosted_models', 'system')
ON CONFLICT (version) DO NOTHING;
```

36.4 REGISTRY SYNC SERVICE

packages/infrastructure/lambda/registry-sync/handler.ts

```typescript
/**
 * RADIANT v4.2.0 - Registry Sync Service
 *
 * Automated synchronization of model registry:
 * - Daily full sync of provider model lists
 * - 5-minute health checks for all providers
 * - Weekly pricing updates
 * - Self-hosted endpoint validation
 */

import { Pool } from 'pg';
import { EventBridgeClient, PutEventsCommand } from '@aws-sdk/client-eventbridge';
import { SageMakerClient, DescribeEndpointCommand } from '@aws-sdk/client-sagemaker';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });
const eventBridge = new EventBridgeClient({});
const sagemaker = new SageMakerClient({});

// ============================================================================
// SYNC TYPES
// ============================================================================

type SyncType = 'full' | 'health' | 'pricing' | 'thermal';

interface SyncResult {
  syncId: string;
  type: SyncType;
  providersUpdated: number;
```

```
  modelsAdded: number;
  modelsUpdated: number;
  modelsDeprecated: number;
  errors: string[];
  durationMs: number;
}

// ============================================================================
// PROVIDER SYNC HANDLERS
// ============================================================================

async function syncProviderModels(providerId: string): Promise<{ added: number; updated: nu
  // Provider-specific model discovery
  switch (providerId) {
    case 'openai':
      return syncOpenAIModels();
    case 'anthropic':
      return syncAnthropicModels();
    case 'google':
      return syncGoogleModels();
    // ... other providers
    default:
      return { added: 0, updated: 0 };
  }
}

async function syncOpenAIModels(): Promise<{ added: number; updated: number }> {
  // OpenAI has a models endpoint
  const response = await fetch('https://api.openai.com/v1/models', {
    headers: { 'Authorization': `Bearer ${process.env.OPENAI_API_KEY}` }
  });

  if (!response.ok) return { added: 0, updated: 0 };

  const data = await response.json();
  let added = 0, updated = 0;

  for (const model of data.data) {
    const existing = await pool.query(
      'SELECT id FROM models WHERE provider_id = $1 AND model_id = $2',
      ['openai', model.id]
    );

    if (existing.rows.length === 0) {
      // New model discovered - flag for admin review
      await pool.query(`
```

```
        INSERT INTO registry_sync_log (sync_type, status, error_message)
        VALUES ('models', 'pending_review', $1)
      `, [`New OpenAI model discovered: ${model.id}`]);
      added++;
    }
  }

  return { added, updated };
}

async function syncAnthropicModels(): Promise<{ added: number; updated: number }> {
  // Anthropic doesn't have a public models endpoint
  // Sync from known model list
  const KNOWN_ANTHROPIC_MODELS = [
    'claude-3-opus-20240229',
    'claude-3-sonnet-20240229',
    'claude-3-haiku-20240307',
    'claude-3-5-sonnet-20241022',
    'claude-opus-4-20250514',
    'claude-sonnet-4-20250514',
  ];

  // Check for any unknown models in our database
  const result = await pool.query(
    'SELECT model_id FROM models WHERE provider_id = $1',
    ['anthropic']
  );

  const knownIds = new Set(KNOWN_ANTHROPIC_MODELS);
  let deprecated = 0;

  for (const row of result.rows) {
    if (!knownIds.has(row.model_id)) {
      // Model may be deprecated
      await pool.query(
        'UPDATE models SET deprecated = true WHERE provider_id = $1 AND model_id = $2',
        ['anthropic', row.model_id]
      );
      deprecated++;
    }
  }

  return { added: 0, updated: deprecated };
}

async function syncGoogleModels(): Promise<{ added: number; updated: number }> {
```

```
  // Google Gemini models
  try {
    const response = await fetch(
      `https://generativelanguage.googleapis.com/v1beta/models?key=${process.env.GOOGLE_API_
    );

    if (!response.ok) return { added: 0, updated: 0 };

    const data = await response.json();
    // Process discovered models...
    return { added: 0, updated: 0 };
  } catch (error) {
    return { added: 0, updated: 0 };
  }
}

// ============================================================================
// HEALTH CHECK HANDLERS
// ============================================================================

async function checkProviderHealth(providerId: string): Promise<void> {
  const provider = await pool.query(
    'SELECT api_base_url FROM providers WHERE id = $1',
    [providerId]
  );

  if (provider.rows.length === 0) return;

  const startTime = Date.now();
  let status = 'healthy';
  let errorMessage: string | null = null;

  try {
    // Simple health check - ping the API
    const response = await fetch(`${provider.rows[0].api_base_url}/models`, {
      method: 'HEAD',
      signal: AbortSignal.timeout(5000)
    });

    if (!response.ok) {
      status = response.status >= 500 ? 'unhealthy' : 'degraded';
    }
  } catch (error: any) {
    status = 'unhealthy';
    errorMessage = error.message;
  }
```

```
  const latencyMs = Date.now() - startTime;

  await pool.query(`
    INSERT INTO provider_health (provider_id, status, avg_latency_ms, last_check_at, last_er
    VALUES ($1, $2, $3, NOW(), $4)
    ON CONFLICT (provider_id, region) DO UPDATE SET
      status = EXCLUDED.status,
      avg_latency_ms = (provider_health.avg_latency_ms * 0.7 + EXCLUDED.avg_latency_ms * 0.3
      last_check_at = NOW(),
      last_success_at = CASE WHEN EXCLUDED.status = 'healthy' THEN NOW() ELSE provider_healt
      last_failure_at = CASE WHEN EXCLUDED.status != 'healthy' THEN NOW() ELSE provider_heal
      last_error = EXCLUDED.last_error,
      updated_at = NOW()
  `, [providerId, status, latencyMs, errorMessage]);
}

async function checkSageMakerEndpoints(): Promise<void> {
  const models = await pool.query(
    'SELECT model_id FROM self_hosted_models WHERE enabled = true'
  );

  for (const model of models.rows) {
    try {
      const endpoint = await sagemaker.send(new DescribeEndpointCommand({
        EndpointName: `radiant-${model.model_id}`
      }));

      const status = endpoint.EndpointStatus === 'InService' ? 'WARM' :
                     endpoint.EndpointStatus === 'Creating' ? 'COLD' : 'OFF';

      await pool.query(`
        UPDATE thermal_states SET
          current_state = $1,
          is_transitioning = $2,
          updated_at = NOW()
        WHERE model_id = $3
      `, [status, endpoint.EndpointStatus === 'Creating', model.model_id]);
    } catch (error) {
      // Endpoint doesn't exist - model is OFF
      await pool.query(`
        UPDATE thermal_states SET
          current_state = 'OFF',
          is_transitioning = false,
          updated_at = NOW()
        WHERE model_id = $1
```

```
      `, [model.model_id]);
    }
  }
}

// ============================================================================
// MAIN SYNC HANDLER
// ============================================================================

export async function handler(event: any): Promise<SyncResult> {
  const syncType: SyncType = event.syncType || 'full';
  const startTime = Date.now();

  // Create sync log entry
  const logResult = await pool.query(`
    INSERT INTO registry_sync_log (sync_type, status)
    VALUES ($1, 'running')
    RETURNING id
  `, [syncType]);
  const syncId = logResult.rows[0].id;

  let providersUpdated = 0;
  let modelsAdded = 0;
  let modelsUpdated = 0;
  let modelsDeprecated = 0;
  const errors: string[] = [];

  try {
    // Get all enabled providers
    const providers = await pool.query(
      'SELECT id FROM providers WHERE enabled = true'
    );

    for (const provider of providers.rows) {
      try {
        switch (syncType) {
          case 'full':
            const result = await syncProviderModels(provider.id);
            modelsAdded += result.added;
            modelsUpdated += result.updated;
            await checkProviderHealth(provider.id);
            providersUpdated++;
            break;

          case 'health':
            await checkProviderHealth(provider.id);
```

```
        providersUpdated++;
        break;

      case 'pricing':
        // Pricing sync - use Section 31 pricing endpoints
        // POST /api/admin/models/{id}/pricing to update
        // await this.syncModelPricing(model.id, pricingData);
        break;
    }
  } catch (error: any) {
    errors.push(`${provider.id}: ${error.message}`);
  }
}

// Check self-hosted endpoints for thermal sync
if (syncType === 'thermal' || syncType === 'full') {
  await checkSageMakerEndpoints();
}

// Refresh materialized view if exists
await pool.query('REFRESH MATERIALIZED VIEW CONCURRENTLY unified_model_stats')
  .catch(() => {}); // Ignore if view doesn't exist

const durationMs = Date.now() - startTime;

// Update sync log
await pool.query(`
  UPDATE registry_sync_log SET
    status = 'completed',
    providers_updated = $1,
    models_added = $2,
    models_updated = $3,
    models_deprecated = $4,
    errors = $5,
    completed_at = NOW(),
    duration_ms = $6
  WHERE id = $7
`, [providersUpdated, modelsAdded, modelsUpdated, modelsDeprecated, errors, durationMs,

// Emit completion event
await eventBridge.send(new PutEventsCommand({
  Entries: [{
    Source: 'radiant.registry',
    DetailType: 'RegistrySyncCompleted',
    Detail: JSON.stringify({
      syncId,
```

```
          syncType,
          providersUpdated,
          modelsAdded,
          modelsUpdated,
          modelsDeprecated,
          durationMs,
          errors
        })
      }]
    }));

    return {
      syncId,
      type: syncType,
      providersUpdated,
      modelsAdded,
      modelsUpdated,
      modelsDeprecated,
      errors,
      durationMs
    };

  } catch (error: any) {
    await pool.query(`
      UPDATE registry_sync_log SET
        status = 'failed',
        error_message = $1,
        completed_at = NOW()
      WHERE id = $2
    `, [error.message, syncId]);

    throw error;
  }
}
```

36.5 CDK INFRASTRUCTURE

packages/infrastructure/lib/stacks/registry-sync-stack.ts

```
/**
 * RADIANT v4.2.0 - Registry Sync CDK Stack
 */

import * as cdk from 'aws-cdk-lib';
import * as lambda from 'aws-cdk-lib/aws-lambda';
```

```typescript
import * as events from 'aws-cdk-lib/aws-events';
import * as targets from 'aws-cdk-lib/aws-events-targets';
import { Construct } from 'constructs';

export interface RegistrySyncStackProps extends cdk.StackProps {
  databaseUrl: string;
  vpcId: string;
}

export class RegistrySyncStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props: RegistrySyncStackProps) {
    super(scope, id, props);

    // Registry Sync Lambda
    const syncLambda = new lambda.Function(this, 'RegistrySyncLambda', {
      functionName: 'radiant-registry-sync',
      runtime: lambda.Runtime.NODEJS_20_X,
      handler: 'handler.handler',
      code: lambda.Code.fromAsset('lambda/registry-sync'),
      timeout: cdk.Duration.minutes(5),
      memorySize: 512,
      environment: {
        DATABASE_URL: props.databaseUrl,
        OPENAI_API_KEY: process.env.OPENAI_API_KEY || '',
        ANTHROPIC_API_KEY: process.env.ANTHROPIC_API_KEY || '',
        GOOGLE_API_KEY: process.env.GOOGLE_API_KEY || '',
      },
    });

    // Daily full sync (00:00 UTC)
    new events.Rule(this, 'DailyFullSync', {
      schedule: events.Schedule.cron({ minute: '0', hour: '0' }),
      targets: [new targets.LambdaFunction(syncLambda, {
        event: events.RuleTargetInput.fromObject({ syncType: 'full' })
      })],
    });

    // Health check every 5 minutes
    new events.Rule(this, 'HealthCheck', {
      schedule: events.Schedule.rate(cdk.Duration.minutes(5)),
      targets: [new targets.LambdaFunction(syncLambda, {
        event: events.RuleTargetInput.fromObject({ syncType: 'health' })
      })],
    });

    // Thermal state sync every 5 minutes
```

```
    new events.Rule(this, 'ThermalSync', {
      schedule: events.Schedule.rate(cdk.Duration.minutes(5)),
      targets: [new targets.LambdaFunction(syncLambda, {
        event: events.RuleTargetInput.fromObject({ syncType: 'thermal' })
      })],
    });

    // Weekly pricing sync (Sunday 00:00 UTC)
    new events.Rule(this, 'WeeklyPricingSync', {
      schedule: events.Schedule.cron({ minute: '0', hour: '0', weekDay: 'SUN' }),
      targets: [new targets.LambdaFunction(syncLambda, {
        event: events.RuleTargetInput.fromObject({ syncType: 'pricing' })
      })],
    });
  }
}
```

36.6 ORCHESTRATION ENGINE MODEL SELECTION

packages/infrastructure/lambda/orchestration/model-selector.ts

```
/**
 * RADIANT v4.2.0 - Orchestration Model Selection
 *
 * Smart model selection using unified registry with:
 * - Thermal state awareness (prefer HOT > WARM > COLD)
 * - Health status filtering
 * - Tier-based eligibility
 * - Capability matching
 */

import { Pool } from 'pg';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });

// =============================================================================
// TYPES
// =============================================================================

export interface ModelSelectionCriteria {
  // Required
  task: 'chat' | 'completion' | 'embedding' | 'image' | 'video' | 'audio' | 'transcription'
  inputModality: string[];
  outputModality: string[];
```

```typescript
  // Tenant context
  tenantTier: 1 | 2 | 3 | 4 | 5;

  // Preferences
  preferHosting?: 'external' | 'self_hosted' | 'any';
  preferProvider?: string[];
  maxLatencyMs?: number;
  maxCostPerRequest?: number;

  // Requirements
  requiredCapabilities?: string[];
  minContextWindow?: number;
  requireHIPAA?: boolean;
}

export interface SelectedModel {
  modelId: string;
  displayName: string;
  hostingType: 'external' | 'self_hosted';
  providerName: string;
  primaryMode: string;
  thermalState: string | null;
  warmupRequired: boolean;
  warmupTimeSeconds: number | null;
  healthStatus: string;
  litellmId: string;
}

// ============================================================================
// MODEL SELECTOR
// ============================================================================

export class ModelSelector {
  async selectModel(criteria: ModelSelectionCriteria): Promise<SelectedModel | null> {
    // Use the database function for initial selection
    const result = await pool.query(`
      SELECT * FROM select_model($1, $2, $3, $4, $5, $6, $7, $8)
    `, [
      criteria.task,
      criteria.inputModality,
      criteria.outputModality,
      criteria.tenantTier,
      criteria.preferHosting || 'any',
      criteria.requiredCapabilities || [],
      criteria.minContextWindow || null,
      criteria.requireHIPAA || false
```

```
    ]);

    if (result.rows.length === 0) {
      return null;
    }

    const selected = result.rows[0];

    // Get full model details
    const modelDetails = await pool.query(`
      SELECT litellm_id FROM unified_model_registry
      WHERE model_id = $1
    `, [selected.model_id]);

    return {
      modelId: selected.model_id,
      displayName: selected.display_name,
      hostingType: selected.hosting_type,
      providerName: selected.provider_name,
      primaryMode: selected.primary_mode,
      thermalState: selected.thermal_state,
      warmupRequired: selected.warmup_required,
      warmupTimeSeconds: selected.warmup_time_seconds,
      healthStatus: selected.health_status || 'unknown',
      litellmId: modelDetails.rows[0]?.litellm_id || selected.model_id
    };
  }

  async selectWithFallback(criteria: ModelSelectionCriteria): Promise<SelectedModel> {
    // Try primary selection
    const primary = await this.selectModel(criteria);
    if (primary && !primary.warmupRequired) {
      return primary;
    }

    // If primary requires warmup, try to find a ready alternative
    if (primary?.warmupRequired) {
      const alternative = await this.selectModel({
        ...criteria,
        preferHosting: 'external' // External providers are always ready
      });

      if (alternative) {
        // Trigger warmup of self-hosted model in background
        this.triggerWarmup(primary.modelId);
        return alternative;
```

```
    }
  }

  // No alternatives - return primary (may require warmup)
  if (primary) {
    return primary;
  }

  // Fallback to default model for task
  return this.getDefaultModel(criteria.task, criteria.tenantTier);
}

private async triggerWarmup(modelId: string): Promise<void> {
  // Trigger warmup via thermal manager
  await pool.query(`
    UPDATE thermal_states SET
      target_state = 'WARM',
      is_transitioning = true,
      updated_at = NOW()
    WHERE model_id = $1 AND current_state = 'COLD'
  `, [modelId]);
}

private async getDefaultModel(task: string, tier: number): Promise<SelectedModel> {
  // Default models by task
  const defaults: Record<string, string> = {
    'chat': 'gpt-4o-mini',
    'completion': 'gpt-4o-mini',
    'embedding': 'text-embedding-3-small',
    'image': 'dall-e-3',
    'video': 'runway-gen3-alpha-turbo',
    'audio': 'tts-1',
    'transcription': 'whisper-1',
    'search': 'perplexity-sonar',
    '3d': 'meshy-v3',
    'inference': 'gpt-4o'
  };

  const modelId = defaults[task] || 'gpt-4o-mini';

  const result = await pool.query(`
    SELECT * FROM unified_model_registry WHERE model_id = $1
  `, [modelId]);

  if (result.rows.length === 0) {
    throw new Error(`Default model ${modelId} not found in registry`);
```

```
    }

    const model = result.rows[0];
    return {
      modelId: model.model_id,
      displayName: model.display_name,
      hostingType: model.hosting_type,
      providerName: model.provider_name,
      primaryMode: model.primary_mode,
      thermalState: model.thermal_state,
      warmupRequired: false,
      warmupTimeSeconds: null,
      healthStatus: model.health_status || 'unknown',
      litellmId: model.litellm_id
    };
  }
}

export const modelSelector = new ModelSelector();
```

36.7 ADMIN API ENDPOINTS

packages/infrastructure/lambda/admin/registry-admin.ts

```
/**
 * RADIANT v4.2.0 - Registry Admin API
 */

import { APIGatewayProxyHandler } from 'aws-lambda';
import { Pool } from 'pg';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });

export const listAllModels: APIGatewayProxyHandler = async (event) => {
  const { category, hostingType, status } = event.queryStringParameters || {};

  let query = 'SELECT * FROM unified_model_registry WHERE 1=1';
  const params: any[] = [];

  if (category) {
    params.push(category);
    query += ` AND category = $${params.length}`;
  }
  if (hostingType) {
    params.push(hostingType);
```

```
    query += ` AND hosting_type = $${params.length}`;
  }
  if (status) {
    params.push(status === 'enabled');
    query += ` AND enabled = $${params.length}`;
  }

  query += ' ORDER BY hosting_type, category, display_name';

  const result = await pool.query(query, params);

  return {
    statusCode: 200,
    body: JSON.stringify({
      total: result.rows.length,
      external: result.rows.filter(r => r.hosting_type === 'external').length,
      selfHosted: result.rows.filter(r => r.hosting_type === 'self_hosted').length,
      models: result.rows
    })
  };
};

export const getRegistryStats: APIGatewayProxyHandler = async () => {
  const stats = await pool.query(`
    SELECT
      COUNT(*) FILTER (WHERE hosting_type = 'external') AS external_count,
      COUNT(*) FILTER (WHERE hosting_type = 'self_hosted') AS self_hosted_count,
      COUNT(*) FILTER (WHERE health_status = 'healthy') AS healthy_count,
      COUNT(*) FILTER (WHERE health_status = 'unhealthy') AS unhealthy_count,
      COUNT(*) FILTER (WHERE thermal_state = 'HOT') AS hot_count,
      COUNT(*) FILTER (WHERE thermal_state = 'WARM') AS warm_count,
      COUNT(*) FILTER (WHERE thermal_state = 'COLD') AS cold_count,
      COUNT(DISTINCT category) AS category_count,
      COUNT(DISTINCT provider_name) AS provider_count
    FROM unified_model_registry
  `);

  return {
    statusCode: 200,
    body: JSON.stringify(stats.rows[0])
  };
};

export const getSyncHistory: APIGatewayProxyHandler = async () => {
  const result = await pool.query(`
    SELECT * FROM registry_sync_log
```

```
    ORDER BY started_at DESC
    LIMIT 50
  `);

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows)
  };
};

export const triggerSync: APIGatewayProxyHandler = async (event) => {
  const { syncType } = JSON.parse(event.body || '{}');

  // Invoke sync lambda
  const lambda = require('@aws-sdk/client-lambda');
  const client = new lambda.LambdaClient({});

  await client.send(new lambda.InvokeCommand({
    FunctionName: 'radiant-registry-sync',
    InvocationType: 'Event',
    Payload: JSON.stringify({ syncType: syncType || 'full' })
  }));

  return {
    statusCode: 202,
    body: JSON.stringify({ message: 'Sync triggered', syncType })
  };
};
```

36.8 VERIFICATION COMMANDS

```
# Apply unified registry migration
psql $DATABASE_URL -f packages/database/migrations/036_unified_model_registry.sql

# Seed self-hosted models
psql $DATABASE_URL -f packages/database/migrations/036a_seed_self_hosted_models.sql

# Verify self-hosted models count (should be 56)
psql $DATABASE_URL -c "SELECT COUNT(*) FROM self_hosted_models"

# Verify unified registry view works
psql $DATABASE_URL -c "SELECT COUNT(*), hosting_type FROM unified_model_registry GROUP BY ho

# Test model selection function
```

```
psql $DATABASE_URL -c "SELECT * FROM select_model('chat', ARRAY['text'], ARRAY['text'], 3, '

# Verify provider health table
psql $DATABASE_URL -c "SELECT provider_id, status, avg_latency_ms FROM provider_health"

# Check sync log
psql $DATABASE_URL -c "SELECT sync_type, status, providers_updated, models_added FROM regist

# Test API endpoints
curl -H "Authorization: Bearer $ADMIN_TOKEN" \
  https://admin-api.example.com/api/v2/admin/registry/models

curl -H "Authorization: Bearer $ADMIN_TOKEN" \
  https://admin-api.example.com/api/v2/admin/registry/stats
```

Section 36 Summary

RADIANT v4.2.0 (PROMPT-16) adds **Unified Model Registry & Sync Service**:

Section 36: Unified Model Registry (v4.2.0)

1. **Database Schema** (036_unified_model_registry.sql)
   - `self_hosted_models` - Complete catalog of 56 SageMaker models
   - `provider_health` - Real-time health monitoring per provider
   - `registry_sync_log` - Sync operation history
   - `unified_model_registry` - SQL VIEW combining ALL 106 models
   - `select_model()` - Smart selection function with thermal awareness
2. **Self-Hosted Model Seed Data** (036a_seed_self_hosted_models.sql)
   - 13 Computer Vision models (EfficientNet, YOLO, SAM, CLIP, etc.)
   - 6 Audio/Speech models (Whisper, TitaNet, pyannote, etc.)
   - 8 Scientific models (AlphaFold 2, ESM-2, RoseTTAFold2, etc.)
   - 6 Medical Imaging models (nnU-Net, MedSAM, CheXNet, etc.)
   - 4 Geospatial models (Prithvi, SatMAE, GeoSAM)
   - 5 3D/Reconstruction models (Nerfstudio, 3DGS, Point-E, etc.)
   - 14 LLM/Embedding models (Llama, Mistral, Qwen, BGE, etc.)
3. **Registry Sync Service** (registry-sync/handler.ts)
   - Daily full sync of provider model lists
   - 5-minute health checks for all providers
   - 5-minute thermal state sync for self-hosted
   - Weekly pricing updates
   - EventBridge events for sync completion
4. **CDK Infrastructure** (registry-sync-stack.ts)
   - Lambda function for sync operations
   - EventBridge rules for scheduled syncs

- IAM permissions for SageMaker access

5. **Model Selector** (model-selector.ts)
   - `selectModel()` - Primary selection with criteria matching
   - `selectWithFallback()` - Fallback to external if warmup needed
   - Thermal state awareness (HOT > WARM > COLD)
   - Health status filtering
6. **Admin API Endpoints**
   - `GET /api/v2/admin/registry/models` - List all models
   - `GET /api/v2/admin/registry/stats` - Registry statistics
   - `GET /api/v2/admin/registry/sync/history` - Sync history
   - `POST /api/v2/admin/registry/sync` - Trigger manual sync

Design Philosophy (v4.2.0)

- **Unified View** - Single source of truth for ALL 106 models
- **hosting_type Field** - Clear 'external' vs 'self_hosted' distinction
- **Automated Sync** - Daily provider sync, 5-min health checks
- **Thermal-Aware** - Prefer ready models, warmup in background
- **Complete Metadata** - Every field needed for orchestration

Also includes all v4.1.0 features:

- Database-Driven Orchestration Engine
- AlphaFold 2 Integration
- License Management & Compliance
- Admin Model CRUD

Also includes all v4.0.0 features:

- Time Machine visual history
- Media Vault with S3 versioning
- Export bundles

Also includes all v3.8.0 features:

- User Model Selection (15 Standard + 15 Novel)
- Admin Editable Pricing
- Cost Transparency per message
- Model Favorites

SECTION-37-FEEDBACK-LEARNING

SECTION 37: FEEDBACK LEARNING SYSTEM & NEURAL ENGINE LOOP (v4.3.0)

Dependencies: Sections 0-36, especially 11 (Brain) and 13 (Neural Engine) **Creates:** Complete feedback loop from user signals → Neural Engine → Brain decisions

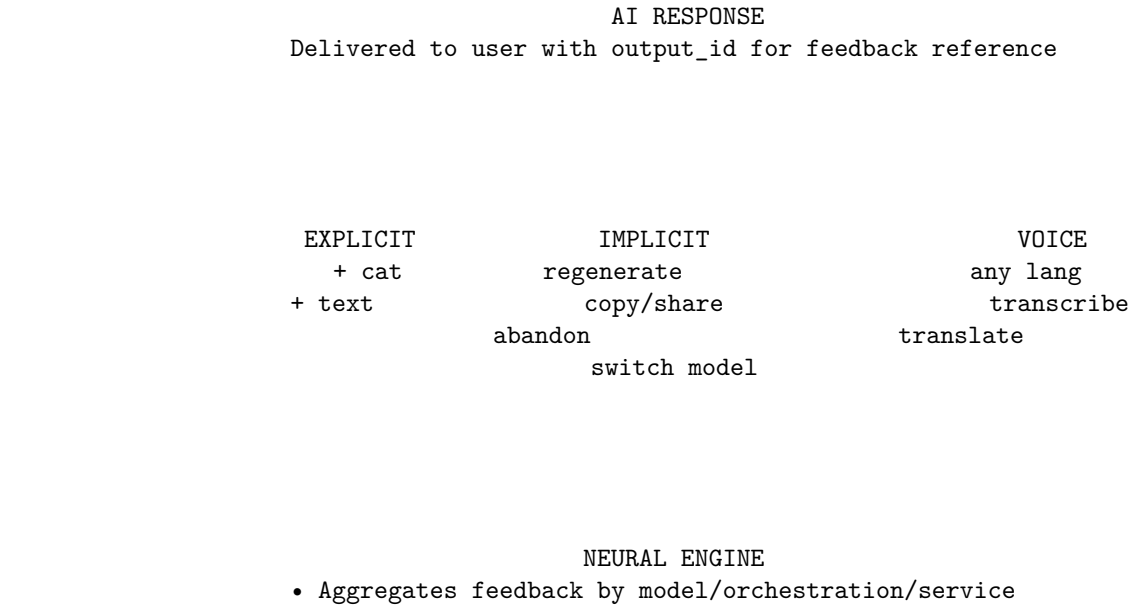
---

37.1 Feedback System Overview

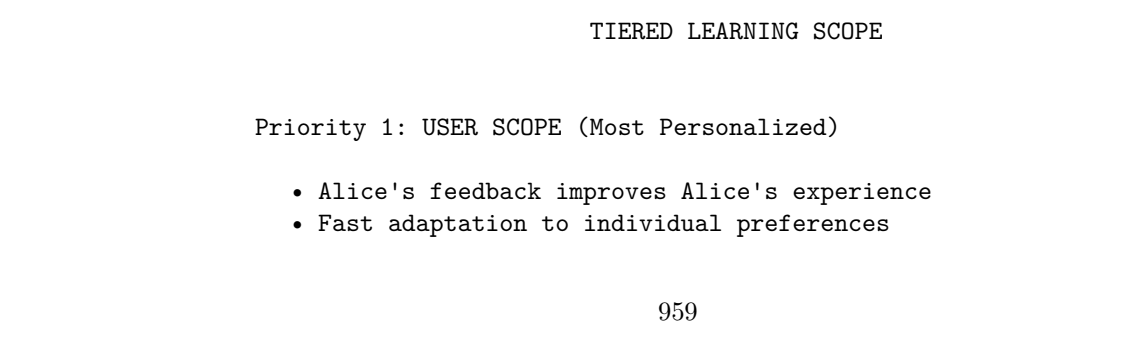
The Feedback Learning System creates a closed-loop where user feedback continuously improves AI routing decisions. The system captures both explicit feedback (thumbs up/down) and implicit signals (regenerate, copy, abandon), ties each to a complete execution manifest, and feeds everything into the Neural Engine which advises the Brain.

Architecture Flow

```
                    FEEDBACK LEARNING LOOP

User Request

                        RADIANT BRAIN
  • Consults Neural Engine for recommendations
  • Considers user/tenant/global learning
  • Applies confidence thresholds
  • Selects orchestration → services → models

                    EXECUTION MANIFEST
Records: output_id, models[], versions[], orchestration_id,
         services[], thermal_states{}, provider_health{},
         brain_reasoning, latency_ms, cost, timestamp
```



Learning Scope Hierarchy



- Requires minimum ~20 feedback samples for confidence

Priority 2: TENANT SCOPE (Organization-Wide)

- Company X's employees collectively train Company X's Brain
- Isolated from Company Y (privacy boundary)
- Captures organizational preferences (e.g., "we prefer Claude")

Priority 3: GLOBAL SCOPE (Platform-Wide, Anonymized)

- Aggregate patterns: "Claude wins 73% for legal tasks"
- Cold start defaults for new users/tenants
- Configurable opt-in/opt-out per tenant

37.2 Feedback Database Schema

```
-- migrations/037_feedback_learning_system.sql

-- ============================================================================
-- EXECUTION MANIFESTS: Full provenance for every AI output
-- ============================================================================

CREATE TABLE execution_manifests (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    output_id VARCHAR(100) UNIQUE NOT NULL,  -- Reference ID for feedback

    -- Context
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    conversation_id UUID REFERENCES thinktank_conversations(id),
    message_id UUID REFERENCES thinktank_messages(id),

    -- What was requested
    request_type VARCHAR(50) NOT NULL,  -- 'chat', 'completion', 'orchestration', 'service'
    task_type VARCHAR(50),  -- 'code', 'creative', 'analysis', 'medical', etc.
    domain_mode VARCHAR(50),  -- Think Tank domain mode if applicable
    input_prompt_hash VARCHAR(64),  -- SHA-256 of input for deduplication
```

```
    input_tokens INTEGER,
    input_language VARCHAR(10),  -- Detected input language (ISO 639-1)

    -- What was used (THE MANIFEST)
    models_used TEXT[] NOT NULL,
    model_versions JSONB DEFAULT '{}',  -- {"claude-sonnet-4": "20241022", ...}
    orchestration_id UUID REFERENCES workflow_definitions(id),
    orchestration_name VARCHAR(100),
    services_used TEXT[],  -- ['perception', 'medical', ...]
    thermal_states_at_execution JSONB DEFAULT '{}',  -- {"whisper-large-v3": "HOT", ...}
    provider_health_at_execution JSONB DEFAULT '{}',  -- {"anthropic": {"latency_ms": 150,

    -- Brain's decision
    brain_reasoning TEXT,  -- Why Brain chose this path
    brain_confidence DECIMAL(3, 2),  -- 0.00-1.00
    was_user_override BOOLEAN DEFAULT false,  -- User manually selected model

    -- Outcome metrics
    output_tokens INTEGER,
    total_latency_ms INTEGER,
    time_to_first_token_ms INTEGER,
    total_cost DECIMAL(10, 6),
    was_streamed BOOLEAN DEFAULT false,

    -- For multi-step orchestrations
    step_count INTEGER DEFAULT 1,
    step_details JSONB DEFAULT '[]',  -- [{model, latency, tokens, cost}, ...]

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT valid_confidence CHECK (brain_confidence >= 0 AND brain_confidence <= 1)
);

CREATE INDEX idx_exec_manifest_output ON execution_manifests(output_id);
CREATE INDEX idx_exec_manifest_tenant_user ON execution_manifests(tenant_id, user_id);
CREATE INDEX idx_exec_manifest_conversation ON execution_manifests(conversation_id);
CREATE INDEX idx_exec_manifest_models ON execution_manifests USING GIN(models_used);
CREATE INDEX idx_exec_manifest_task ON execution_manifests(task_type);
CREATE INDEX idx_exec_manifest_created ON execution_manifests(created_at DESC);

-- ============================================================================
-- EXPLICIT FEEDBACK: Thumbs up/down with optional categories
-- ============================================================================

CREATE TYPE feedback_rating AS ENUM ('positive', 'negative', 'neutral');
CREATE TYPE feedback_category AS ENUM (
```

```
    'accuracy',      -- Factually correct
    'relevance',     -- Answered the question
    'tone',          -- Appropriate style
    'format',        -- Good structure/formatting
    'speed',         -- Fast enough
    'safety',        -- Appropriate content
    'creativity',    -- Novel/interesting
    'completeness',  -- Fully addressed request
    'other'
);

CREATE TABLE feedback_explicit (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Link to execution
    output_id VARCHAR(100) NOT NULL REFERENCES execution_manifests(output_id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- The feedback
    rating feedback_rating NOT NULL,
    categories feedback_category[] DEFAULT '{}',  -- What specifically was good/bad
    comment_text TEXT,  -- Optional text feedback
    comment_language VARCHAR(10),  -- Detected language of comment

    -- Metadata
    feedback_source VARCHAR(50) DEFAULT 'thinktank',  -- 'thinktank', 'api', 'admin'
    user_agent TEXT,
    client_timestamp TIMESTAMPTZ,  -- When user clicked

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    -- One feedback per output per user
    UNIQUE(output_id, user_id)
);

CREATE INDEX idx_feedback_explicit_output ON feedback_explicit(output_id);
CREATE INDEX idx_feedback_explicit_tenant ON feedback_explicit(tenant_id);
CREATE INDEX idx_feedback_explicit_rating ON feedback_explicit(rating);
CREATE INDEX idx_feedback_explicit_created ON feedback_explicit(created_at DESC);

-- ============================================================================
-- IMPLICIT FEEDBACK: Behavioral signals (higher volume, weighted less)
-- ============================================================================

CREATE TYPE implicit_signal_type AS ENUM (
```

```
    'regenerate',           -- User clicked regenerate (negative)
    'copy_response',        -- User copied response (positive)
    'share_response',       -- User shared response (very positive)
    'export_response',      -- User exported (positive)
    'continue_conversation',-- User sent follow-up (neutral-positive)
    'abandon_conversation', -- User left without follow-up (weak negative)
    'abandon_mid_response', -- User stopped streaming (negative)
    'manual_model_switch',  -- User changed model (negative for current)
    'edit_and_resend',      -- User edited their prompt (neutral)
    'response_time_short',  -- Quick reply = engaged (positive)
    'response_time_long',   -- Slow reply = confused/distracted (neutral)
    'scroll_to_end',        -- Read full response (positive)
    'scroll_bounce',        -- Scrolled away quickly (negative)
    'favorite_added',       -- Added to favorites (very positive)
    'report_content'        -- Reported for review (very negative)
);

CREATE TABLE feedback_implicit (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Link to execution
    output_id VARCHAR(100) NOT NULL REFERENCES execution_manifests(output_id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- The signal
    signal_type implicit_signal_type NOT NULL,
    signal_value JSONB DEFAULT '{}',  -- Additional context (e.g., time_to_next_message_ms)

    -- Computed sentiment score (-1.0 to +1.0)
    sentiment_score DECIMAL(3, 2) NOT NULL,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_feedback_implicit_output ON feedback_implicit(output_id);
CREATE INDEX idx_feedback_implicit_tenant ON feedback_implicit(tenant_id);
CREATE INDEX idx_feedback_implicit_signal ON feedback_implicit(signal_type);
CREATE INDEX idx_feedback_implicit_created ON feedback_implicit(created_at DESC);

-- ============================================================================
-- VOICE FEEDBACK: Multi-language voice input with transcription
-- ============================================================================

CREATE TABLE feedback_voice (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    -- Link to execution
    output_id VARCHAR(100) NOT NULL REFERENCES execution_manifests(output_id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- Audio storage
    audio_s3_key VARCHAR(500) NOT NULL,
    audio_duration_seconds DECIMAL(8, 2),
    audio_format VARCHAR(20),  -- 'webm', 'mp3', 'wav', etc.

    -- Transcription
    transcription_text TEXT,
    original_language VARCHAR(10),  -- Detected language (ISO 639-1)
    translated_text TEXT,  -- English translation if not English
    transcription_confidence DECIMAL(3, 2),
    transcription_model VARCHAR(50),  -- 'whisper-large-v3', etc.

    -- Sentiment analysis of voice feedback
    sentiment_score DECIMAL(3, 2),  -- -1.0 to +1.0
    inferred_rating feedback_rating,
    inferred_categories feedback_category[],

    -- Processing status
    processing_status VARCHAR(20) DEFAULT 'pending',  -- 'pending', 'processing', 'complete
    processing_error TEXT,
    processed_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_feedback_voice_output ON feedback_voice(output_id);
CREATE INDEX idx_feedback_voice_status ON feedback_voice(processing_status);

-- ============================================================================
-- NEURAL MODEL SCORES: Learned effectiveness per model/task/scope
-- ============================================================================

CREATE TYPE learning_scope AS ENUM ('user', 'tenant', 'global');

CREATE TABLE neural_model_scores (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Scope (null = global)
    scope learning_scope NOT NULL,
    tenant_id UUID REFERENCES tenants(id),  -- null for global
```

```
    user_id UUID REFERENCES users(id),      -- null for tenant/global

    -- What we're scoring
    model_id VARCHAR(100) NOT NULL,
    task_type VARCHAR(50),  -- null = overall score for model
    domain_mode VARCHAR(50),  -- null = all domains

    -- Aggregated scores (0.0 to 1.0)
    effectiveness_score DECIMAL(4, 3) NOT NULL DEFAULT 0.500,
    accuracy_score DECIMAL(4, 3),
    relevance_score DECIMAL(4, 3),
    speed_score DECIMAL(4, 3),

    -- Statistics
    positive_count INTEGER DEFAULT 0,
    negative_count INTEGER DEFAULT 0,
    neutral_count INTEGER DEFAULT 0,
    implicit_positive_count INTEGER DEFAULT 0,
    implicit_negative_count INTEGER DEFAULT 0,
    total_feedback_count INTEGER DEFAULT 0,

    -- Confidence in this score (based on sample size)
    confidence DECIMAL(3, 2) DEFAULT 0.00,

    -- Model version tracking
    last_model_version VARCHAR(50),
    score_decay_applied_at TIMESTAMPTZ,  -- When we last decayed old scores

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    -- Unique per scope/model/task combination
    UNIQUE NULLS NOT DISTINCT (scope, tenant_id, user_id, model_id, task_type, domain_mode)
);

CREATE INDEX idx_neural_scores_model ON neural_model_scores(model_id);
CREATE INDEX idx_neural_scores_scope ON neural_model_scores(scope);
CREATE INDEX idx_neural_scores_tenant ON neural_model_scores(tenant_id);
CREATE INDEX idx_neural_scores_effectiveness ON neural_model_scores(effectiveness_score DESC

-- ============================================================================
-- NEURAL ROUTING RECOMMENDATIONS: Brain reads these for decisions
-- ============================================================================

CREATE TABLE neural_routing_recommendations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    -- Scope
    scope learning_scope NOT NULL,
    tenant_id UUID REFERENCES tenants(id),
    user_id UUID REFERENCES users(id),

    -- Recommendation context
    task_type VARCHAR(50) NOT NULL,
    domain_mode VARCHAR(50),
    input_characteristics JSONB DEFAULT '{}',  -- {requires_vision, requires_audio, token_es

    -- The recommendation
    recommended_model VARCHAR(100) NOT NULL,
    recommended_orchestration_id UUID REFERENCES workflow_definitions(id),
    recommended_services TEXT[],

    -- Alternatives (ordered by score)
    alternative_models TEXT[],

    -- Confidence
    recommendation_confidence DECIMAL(3, 2) NOT NULL,
    sample_size INTEGER,  -- How much data this is based on

    -- Reasoning (for debugging/transparency)
    reasoning TEXT,

    -- Validity
    valid_from TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    valid_until TIMESTAMPTZ,  -- null = no expiry
    is_active BOOLEAN DEFAULT true,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_neural_rec_active ON neural_routing_recommendations(is_active, scope);
CREATE INDEX idx_neural_rec_task ON neural_routing_recommendations(task_type, domain_mode);
CREATE INDEX idx_neural_rec_tenant ON neural_routing_recommendations(tenant_id);

-- ============================================================================
-- USER TRUST SCORES: Anti-gaming feedback weighting
-- ============================================================================

CREATE TABLE user_trust_scores (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
```

```
    user_id UUID NOT NULL REFERENCES users(id),

    -- Trust factors
    trust_score DECIMAL(3, 2) NOT NULL DEFAULT 0.50,  -- 0.00-1.00
    account_age_days INTEGER,
    total_feedback_count INTEGER DEFAULT 0,
    feedback_diversity_score DECIMAL(3, 2),  -- How varied their feedback is
    feedback_alignment_score DECIMAL(3, 2),  -- How aligned with population

    -- Flags
    is_outlier BOOLEAN DEFAULT false,
    outlier_reason TEXT,
    is_rate_limited BOOLEAN DEFAULT false,
    rate_limit_until TIMESTAMPTZ,

    -- Last update
    last_feedback_at TIMESTAMPTZ,
    last_trust_calculation_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, user_id)
);

CREATE INDEX idx_trust_scores_tenant ON user_trust_scores(tenant_id);
CREATE INDEX idx_trust_scores_outlier ON user_trust_scores(is_outlier);

-- ============================================================================
-- A/B TESTING: Measure if routing changes improve outcomes
-- ============================================================================

CREATE TYPE experiment_status AS ENUM ('draft', 'running', 'paused', 'completed', 'cancelled

CREATE TABLE ab_experiments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Experiment definition
    name VARCHAR(200) NOT NULL,
    description TEXT,
    hypothesis TEXT,

    -- Scope
    tenant_id UUID REFERENCES tenants(id),  -- null = all tenants

    -- What we're testing
```

```
    experiment_type VARCHAR(50) NOT NULL,  -- 'model_routing', 'orchestration', 'service'
    control_config JSONB NOT NULL,  -- The current/default behavior
    treatment_config JSONB NOT NULL,  -- The new behavior being tested

    -- Traffic allocation
    traffic_percentage DECIMAL(5, 2) DEFAULT 10.00,  -- % of users in treatment

    -- Status
    status experiment_status NOT NULL DEFAULT 'draft',
    started_at TIMESTAMPTZ,
    ended_at TIMESTAMPTZ,

    -- Results
    control_sample_size INTEGER DEFAULT 0,
    treatment_sample_size INTEGER DEFAULT 0,
    control_positive_rate DECIMAL(5, 4),
    treatment_positive_rate DECIMAL(5, 4),
    statistical_significance DECIMAL(5, 4),  -- p-value
    effect_size DECIMAL(5, 4),  -- Cohen's d
    winner VARCHAR(20),  -- 'control', 'treatment', 'inconclusive'

    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_ab_experiments_status ON ab_experiments(status);
CREATE INDEX idx_ab_experiments_tenant ON ab_experiments(tenant_id);

CREATE TABLE ab_experiment_assignments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    experiment_id UUID NOT NULL REFERENCES ab_experiments(id) ON DELETE CASCADE,
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- Assignment
    variant VARCHAR(20) NOT NULL,  -- 'control' or 'treatment'
    assigned_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(experiment_id, user_id)
);

CREATE INDEX idx_ab_assignments_experiment ON ab_experiment_assignments(experiment_id);
CREATE INDEX idx_ab_assignments_user ON ab_experiment_assignments(user_id);

-- ============================================================================
```

```sql
-- ROW LEVEL SECURITY
-- ============================================================================

ALTER TABLE execution_manifests ENABLE ROW LEVEL SECURITY;
ALTER TABLE feedback_explicit ENABLE ROW LEVEL SECURITY;
ALTER TABLE feedback_implicit ENABLE ROW LEVEL SECURITY;
ALTER TABLE feedback_voice ENABLE ROW LEVEL SECURITY;
ALTER TABLE neural_model_scores ENABLE ROW LEVEL SECURITY;
ALTER TABLE neural_routing_recommendations ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_trust_scores ENABLE ROW LEVEL SECURITY;
ALTER TABLE ab_experiments ENABLE ROW LEVEL SECURITY;
ALTER TABLE ab_experiment_assignments ENABLE ROW LEVEL SECURITY;

CREATE POLICY exec_manifest_isolation ON execution_manifests
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY feedback_explicit_isolation ON feedback_explicit
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY feedback_implicit_isolation ON feedback_implicit
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY feedback_voice_isolation ON feedback_voice
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY neural_scores_isolation ON neural_model_scores
    USING (tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY neural_rec_isolation ON neural_routing_recommendations
    USING (tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY trust_scores_isolation ON user_trust_scores
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY ab_experiments_isolation ON ab_experiments
    USING (tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY ab_assignments_isolation ON ab_experiment_assignments
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

-- ============================================================================
-- HELPER FUNCTIONS
-- ============================================================================

-- Get aggregated feedback score for an output
CREATE OR REPLACE FUNCTION get_feedback_score(p_output_id VARCHAR)
RETURNS JSONB AS $$
DECLARE
    explicit_score DECIMAL;
    implicit_score DECIMAL;
    combined_score DECIMAL;
    result JSONB;
BEGIN
    -- Get explicit feedback
```

```
    SELECT
        CASE rating
            WHEN 'positive' THEN 1.0
            WHEN 'negative' THEN -1.0
            ELSE 0.0
        END INTO explicit_score
    FROM feedback_explicit
    WHERE output_id = p_output_id
    LIMIT 1;

    -- Get average implicit feedback
    SELECT AVG(sentiment_score) INTO implicit_score
    FROM feedback_implicit
    WHERE output_id = p_output_id;

    -- Combine (explicit weighted 3x)
    combined_score := COALESCE(
        (COALESCE(explicit_score, 0) * 3 + COALESCE(implicit_score, 0)) /
        CASE
            WHEN explicit_score IS NOT NULL AND implicit_score IS NOT NULL THEN 4
            WHEN explicit_score IS NOT NULL THEN 3
            WHEN implicit_score IS NOT NULL THEN 1
            ELSE 1
        END,
        0
    );

    result := jsonb_build_object(
        'explicit_score', explicit_score,
        'implicit_score', implicit_score,
        'combined_score', combined_score,
        'has_explicit', explicit_score IS NOT NULL,
        'has_implicit', implicit_score IS NOT NULL
    );

    RETURN result;
END;
$$ LANGUAGE plpgsql;

-- Get best model recommendation for context
CREATE OR REPLACE FUNCTION get_neural_recommendation(
    p_tenant_id UUID,
    p_user_id UUID,
    p_task_type VARCHAR,
    p_domain_mode VARCHAR DEFAULT NULL
)
```

```
RETURNS TABLE (
    model_id VARCHAR,
    confidence DECIMAL,
    scope learning_scope,
    reasoning TEXT
) AS $$
BEGIN
    -- Try user-specific first, then tenant, then global
    RETURN QUERY
    SELECT
        r.recommended_model,
        r.recommendation_confidence,
        r.scope,
        r.reasoning
    FROM neural_routing_recommendations r
    WHERE r.is_active = true
    AND (r.valid_until IS NULL OR r.valid_until > NOW())
    AND r.task_type = p_task_type
    AND (r.domain_mode IS NULL OR r.domain_mode = p_domain_mode)
    AND (
        (r.scope = 'user' AND r.tenant_id = p_tenant_id AND r.user_id = p_user_id) OR
        (r.scope = 'tenant' AND r.tenant_id = p_tenant_id AND r.user_id IS NULL) OR
        (r.scope = 'global' AND r.tenant_id IS NULL AND r.user_id IS NULL)
    )
    ORDER BY
        CASE r.scope
            WHEN 'user' THEN 1
            WHEN 'tenant' THEN 2
            WHEN 'global' THEN 3
        END,
        r.recommendation_confidence DESC
    LIMIT 1;
END;
$$ LANGUAGE plpgsql;

-- Calculate implicit signal sentiment
CREATE OR REPLACE FUNCTION get_implicit_sentiment(p_signal_type implicit_signal_type)
RETURNS DECIMAL AS $$
BEGIN
    RETURN CASE p_signal_type
        WHEN 'copy_response' THEN 0.7
        WHEN 'share_response' THEN 0.9
        WHEN 'export_response' THEN 0.6
        WHEN 'favorite_added' THEN 0.9
        WHEN 'continue_conversation' THEN 0.3
        WHEN 'scroll_to_end' THEN 0.2
```

```
        WHEN 'response_time_short' THEN 0.2
        WHEN 'regenerate' THEN -0.8
        WHEN 'manual_model_switch' THEN -0.6
        WHEN 'abandon_conversation' THEN -0.3
        WHEN 'abandon_mid_response' THEN -0.7
        WHEN 'scroll_bounce' THEN -0.4
        WHEN 'report_content' THEN -1.0
        WHEN 'edit_and_resend' THEN 0.0
        WHEN 'response_time_long' THEN 0.0
        ELSE 0.0
    END;
END;
$$ LANGUAGE plpgsql IMMUTABLE;
```

37.3 Shared Types for Feedback System

```typescript
// packages/shared/src/types/feedback.types.ts

/**
 * RADIANT v4.3.0 - Feedback System Types
 */

// =============================================================================
// EXECUTION MANIFEST
// =============================================================================

export interface ExecutionManifest {
    id: string;
    outputId: string;  // Unique reference for feedback

    // Context
    tenantId: string;
    userId: string;
    conversationId?: string;
    messageId?: string;

    // Request
    requestType: 'chat' | 'completion' | 'orchestration' | 'service';
    taskType?: TaskType;
    domainMode?: DomainMode;
    inputPromptHash?: string;
    inputTokens?: number;
    inputLanguage?: string;
```

```
    // THE MANIFEST - What was used
    modelsUsed: string[];
    modelVersions: Record<string, string>;
    orchestrationId?: string;
    orchestrationName?: string;
    servicesUsed: string[];
    thermalStatesAtExecution: Record<string, ThermalState>;
    providerHealthAtExecution: Record<string, ProviderHealthSnapshot>;

    // Brain decision
    brainReasoning?: string;
    brainConfidence: number;
    wasUserOverride: boolean;

    // Outcome
    outputTokens?: number;
    totalLatencyMs: number;
    timeToFirstTokenMs?: number;
    totalCost: number;
    wasStreamed: boolean;

    // Multi-step
    stepCount: number;
    stepDetails: ExecutionStep[];

    createdAt: Date;
}

export interface ExecutionStep {
    stepIndex: number;
    modelId: string;
    latencyMs: number;
    inputTokens: number;
    outputTokens: number;
    cost: number;
}

export interface ProviderHealthSnapshot {
    latencyMs: number;
    errorRate: number;
    status: 'healthy' | 'degraded' | 'unhealthy';
}

export type TaskType =
    | 'chat'
    | 'code'
```

```
    | 'analysis'
    | 'creative'
    | 'vision'
    | 'audio'
    | 'medical'
    | 'legal'
    | 'research'
    | 'translation';

export type DomainMode =
    | 'general'
    | 'medical'
    | 'legal'
    | 'code'
    | 'creative'
    | 'research'
    | 'business';

// ============================================================================
// EXPLICIT FEEDBACK
// ============================================================================

export type FeedbackRating = 'positive' | 'negative' | 'neutral';

export type FeedbackCategory =
    | 'accuracy'
    | 'relevance'
    | 'tone'
    | 'format'
    | 'speed'
    | 'safety'
    | 'creativity'
    | 'completeness'
    | 'other';

export interface ExplicitFeedback {
    id: string;
    outputId: string;
    tenantId: string;
    userId: string;

    rating: FeedbackRating;
    categories: FeedbackCategory[];
    commentText?: string;
    commentLanguage?: string;
```

```
    feedbackSource: 'thinktank' | 'api' | 'admin';
    createdAt: Date;
}

export interface FeedbackSubmission {
    outputId: string;
    rating: FeedbackRating;
    categories?: FeedbackCategory[];
    commentText?: string;
}

// ============================================================================
// IMPLICIT FEEDBACK
// ============================================================================

export type ImplicitSignalType =
    | 'regenerate'
    | 'copy_response'
    | 'share_response'
    | 'export_response'
    | 'continue_conversation'
    | 'abandon_conversation'
    | 'abandon_mid_response'
    | 'manual_model_switch'
    | 'edit_and_resend'
    | 'response_time_short'
    | 'response_time_long'
    | 'scroll_to_end'
    | 'scroll_bounce'
    | 'favorite_added'
    | 'report_content';

export interface ImplicitFeedback {
    id: string;
    outputId: string;
    tenantId: string;
    userId: string;

    signalType: ImplicitSignalType;
    signalValue: Record<string, unknown>;
    sentimentScore: number;  // -1.0 to +1.0

    createdAt: Date;
}

export interface ImplicitSignalSubmission {
```

```
    outputId: string;
    signalType: ImplicitSignalType;
    signalValue?: Record<string, unknown>;
}

// ============================================================================
// VOICE FEEDBACK
// ============================================================================

export interface VoiceFeedback {
    id: string;
    outputId: string;
    tenantId: string;
    userId: string;

    audioS3Key: string;
    audioDurationSeconds: number;
    audioFormat: string;

    transcriptionText?: string;
    originalLanguage?: string;
    translatedText?: string;
    transcriptionConfidence?: number;
    transcriptionModel?: string;

    sentimentScore?: number;
    inferredRating?: FeedbackRating;
    inferredCategories?: FeedbackCategory[];

    processingStatus: 'pending' | 'processing' | 'completed' | 'failed';
    processedAt?: Date;

    createdAt: Date;
}

// ============================================================================
// NEURAL LEARNING
// ============================================================================

export type LearningScope = 'user' | 'tenant' | 'global';

export interface NeuralModelScore {
    id: string;
    scope: LearningScope;
    tenantId?: string;
    userId?: string;
```

```
    modelId: string;
    taskType?: TaskType;
    domainMode?: DomainMode;

    effectivenessScore: number;  // 0.0-1.0
    accuracyScore?: number;
    relevanceScore?: number;
    speedScore?: number;

    positiveCount: number;
    negativeCount: number;
    neutralCount: number;
    implicitPositiveCount: number;
    implicitNegativeCount: number;
    totalFeedbackCount: number;

    confidence: number;  // 0.0-1.0

    updatedAt: Date;
}

export interface NeuralRecommendation {
    id: string;
    scope: LearningScope;
    tenantId?: string;
    userId?: string;

    taskType: TaskType;
    domainMode?: DomainMode;
    inputCharacteristics: Record<string, unknown>;

    recommendedModel: string;
    recommendedOrchestrationId?: string;
    recommendedServices: string[];
    alternativeModels: string[];

    recommendationConfidence: number;
    sampleSize: number;
    reasoning: string;

    isActive: boolean;
    validFrom: Date;
    validUntil?: Date;
}
```

```typescript
// ============================================================================
// TRUST & ANTI-GAMING
// ============================================================================

export interface UserTrustScore {
    id: string;
    tenantId: string;
    userId: string;

    trustScore: number;  // 0.0-1.0
    accountAgeDays: number;
    totalFeedbackCount: number;
    feedbackDiversityScore: number;
    feedbackAlignmentScore: number;

    isOutlier: boolean;
    outlierReason?: string;
    isRateLimited: boolean;
    rateLimitUntil?: Date;

    updatedAt: Date;
}

// ============================================================================
// A/B TESTING
// ============================================================================

export type ExperimentStatus = 'draft' | 'running' | 'paused' | 'completed' | 'cancelled';

export interface ABExperiment {
    id: string;
    name: string;
    description?: string;
    hypothesis?: string;

    tenantId?: string;
    experimentType: 'model_routing' | 'orchestration' | 'service';
    controlConfig: Record<string, unknown>;
    treatmentConfig: Record<string, unknown>;

    trafficPercentage: number;
    status: ExperimentStatus;

    controlSampleSize: number;
    treatmentSampleSize: number;
    controlPositiveRate?: number;
```

```
    treatmentPositiveRate?: number;
    statisticalSignificance?: number;
    effectSize?: number;
    winner?: 'control' | 'treatment' | 'inconclusive';

    startedAt?: Date;
    endedAt?: Date;
}

// Import existing types
import { ThermalState } from './ai.types';
```

Update the shared types index:

```
// packages/shared/src/types/index.ts

// Add to existing exports
export * from './feedback.types';
```

37.4 API Endpoints Summary

Feedback Endpoints

Method	Endpoint	Description
POST	`/api/v2/feedback/explicit`	Submit thumbs up/down with optional categories
POST	`/api/v2/feedback/implicit`	Record implicit signal (regenerate, copy, etc.)
POST	`/api/v2/feedback/voice`	Upload voice feedback (multipart)
GET	`/api/v2/feedback/stats/:outputId`	Get aggregated feedback for output

Neural Engine Endpoints

Method	Endpoint	Description
GET	`/api/v2/neural/recommendation`	Get Neural Engine recommendation
GET	`/api/v2/neural/scores`	Get model scores for context
POST	`/api/v2/neural/learn`	Trigger learning for output (internal)

Voice Processing Endpoints

Method	Endpoint	Description
POST	`/api/v2/voice/transcribe`	Transcribe audio to text (any language)
POST	`/api/v2/voice/translate`	Translate text to English

Service Layer (Client Apps)

Method	Endpoint	Description
GET	`/api/v2/service/feedback/config`	Get feedback widget configuration
POST	`/api/v2/service/feedback/implicit`	Batch submit implicit signals
GET	`/api/v2/service/feedback/conversation/{id}/summary`	Get conversation feedback summary

Admin Endpoints

Method	Endpoint	Description
GET	`/api/v2/admin/feedback/stats`	Feedback analytics dashboard
GET	`/api/v2/admin/experiments`	List A/B experiments
POST	`/api/v2/admin/experiments`	Create A/B experiment
PUT	`/api/v2/admin/experiments/{id}`	Update experiment status
GET	`/api/v2/admin/trust-scores`	View user trust scores

37.5 Implicit Signal Sentiment Mapping

Signal Type	Sentiment Score	Interpretation
`favorite_added`	+0.9	Very positive - user values this response
`share_response`	+0.9	Very positive - worth sharing
`copy_response`	+0.7	Positive - useful enough to copy
`export_response`	+0.6	Positive - keeping for later
`continue_conversation`	+0.3	Neutral-positive - engaged

Signal Type	Sentiment Score	Interpretation
`scroll_to_end`	+0.2	Weak positive - read full response
`response_time_short`	+0.2	Weak positive - quick engagement
`edit_and_resend`	0.0	Neutral - refining question
`response_time_long`	0.0	Neutral - thinking/distracted
`abandon_conversation`	-0.3	Weak negative - may be done or frustrated
`scroll_bounce`	-0.4	Negative - didn't read response
`manual_model_switch`	-0.6	Negative - current model wasn't working
`abandon_mid_response`	-0.7	Negative - stopped streaming early
`regenerate`	-0.8	Negative - response wasn't good
`report_content`	-1.0	Very negative - safety/quality issue

37.6 Learning Weighting Formula

Combined feedback score for model scoring:

```
weighted_score = (
    (explicit_score × 3.0 × trust_weight) +
    (implicit_score × 1.0) +
    (voice_score × 2.0 × trust_weight)
) / total_weight
```

Where: - `explicit_score`: -1.0 to +1.0 from thumbs rating - `implicit_score`: Average of implicit signal sentiments - `voice_score`: Sentiment analysis of transcribed voice feedback - `trust_weight`: 0.1 to 1.0 based on user trust score

Model effectiveness update:

```
new_score = (current_score × current_count + weighted_score) / (current_count + 1)
confidence = min(total_count / min_sample_size, 1.0)
```

Summary

RADIANT v4.3.0 (PROMPT-17) adds **Feedback Learning System & Neural Engine Loop**:

Section 37: Feedback Learning System (v4.3.0)

1. **Database Schema** (037_feedback_learning_system.sql)
   - `execution_manifests` - Full provenance for every AI output
   - `feedback_explicit` - Thumbs up/down with categories
   - `feedback_implicit` - Behavioral signals (regenerate, copy, abandon, etc.)
   - `feedback_voice` - Multi-language voice feedback with transcription
   - `neural_model_scores` - Learned effectiveness per model/task/scope
   - `neural_routing_recommendations` - Brain advice from Neural Engine
   - `user_trust_scores` - Anti-gaming trust levels
   - `ab_experiments` - A/B testing experiments
2. **Learning Scopes**
   - User-level: Personal preferences, fastest adaptation
   - Tenant-level: Organization-wide patterns, privacy isolated
   - Global-level: Platform defaults, cold start handling
3. **Signal Types**
   - Explicit: Thumbs up/down + 8 feedback categories + text comments
   - Implicit: 15 behavioral signals (regenerate, copy, share, abandon, etc.)
   - Voice: Multi-language voice feedback with Whisper transcription
4. **Neural Engine Integration**
   - Brain consults Neural Engine before every routing decision
   - Neural scores weighted at 35% in model selection
   - Real-time + nightly batch learning
5. **Anti-Gaming Protection**
   - User trust scores (0.0-1.0)
   - Rate limiting (50 feedback/hour)
   - Outlier detection (deviation from population)
   - Account age weighting
6. **A/B Testing Framework**
   - Create experiments comparing routing strategies
   - Automatic user assignment to control/treatment
   - Statistical significance tracking

SECTION-38-NEURAL-ORCHESTRATION

SECTION 38: NEURAL-FIRST ORCHESTRATION & THINK TANK WORKFLOW REGISTRY (v4.4.0)

This section transforms RADIANT from template-based to neural-first orchestration Includes: Think Tank Workflow Registry, Visual Editor, Per-User Neural Models, Real-Time Steering

---

38.1 Concurrent Execution Architecture

CRITICAL: RADIANT supports concurrent execution per user across all systems.

Concurrent Execution Principles

1. **Per-User Parallelism**: Each user can run multiple AI conversations/workflows simultaneously
2. **Billing Awareness**: Usage tracking and cost accumulation work across parallel sessions
3. **Feedback Aggregation**: Feedback from concurrent sessions properly attributed and aggregated
4. **Neural Learning**: Learning signals from parallel sessions contribute to user model without race conditions
5. **Session Isolation**: Each concurrent session has independent state while sharing user preferences

Concurrent Session Architecture

```
                    CONCURRENT EXECUTION PER USER

User A
   Session 1 (Chat)
   Session 2 (Workflow)
   Session 3 (Research)
   Session 4 (Code)
```

CONCURRENT SESSION MANAGER

- Session state isolation
- Independent execution
- Parallel model calls
- Per-session manifests
- Shared user preferences
- Aggregated usage tracking
- Unified feedback collection
- Combined cost accumulation

Billing Service

- Per-session cost tracking
- Aggregated invoicing

Neural Engine

- Atomic updates to user model
- Lock-free learning

Feedback System

- Session-tagged feedback
- Batch learning aggregation

38.2 Section Overview

What This Section Creates

1. **Think Tank Workflow Registry** - 127 orchestration patterns, 127 production workflows, 834 domains in database
2. **Neural-First Architecture** - Neural Engine as fabric, Brain as governor, tight integration loop
3. **Per-User Neural Models** - Personalized embeddings for preferences, domains, behavior
4. **Orchestration Constructor** - Dynamic workflow generation from primitives (not just template selection)
5. **Real-Time Steering** - Neural Engine monitors and adjusts during execution
6. **Visual Workflow Editor** - Comprehensive admin interface for building orchestrations
7. **Client Decision Transparency** - Think Tank receives reasoning, confidence, alternatives
8. **Swift Deployer v2** - Enhanced deployment app with workflow and Neural configuration
9. **Concurrent Execution Support** - Full awareness in billing, feedback, and learning systems

Design Philosophy

Principle	Current (v4.3)	New (v4.4)
Neural Role	35% weight advisor	60% weight fabric with Brain veto
Orchestration	Template selection	Dynamic construction from primitives
User Model	Global defaults	Per-user embeddings + preferences
Feedback	Post-hoc batch learning	Real-time steering + continuous learning
Admin Control	Model selection only	Full parameter visibility and editing
Client Transparency	Result only	Reasoning + confidence + alternatives
Concurrent Support	Implicit	Explicit per-session tracking

38.3 Database Schema: Think Tank Workflow Registry

migrations/038_neural_orchestration_registry.sql

```sql
-- ============================================================================
-- RADIANT v4.4.0 - Neural-First Orchestration & Think Tank Workflow Registry
-- ============================================================================

-- Enable pgvector if not already enabled
CREATE EXTENSION IF NOT EXISTS vector;

-- ============================================================================
-- PART 1: THINK TANK ORCHESTRATION PATTERNS (127 patterns)
-- ============================================================================

-- Orchestration pattern categories
CREATE TABLE IF NOT EXISTS orchestration_pattern_categories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    category_id VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    display_order INTEGER DEFAULT 0,
    pattern_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

```sql
-- Core orchestration patterns (127 total from Think Tank Compendium)
CREATE TABLE IF NOT EXISTS orchestration_patterns (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pattern_id VARCHAR(100) UNIQUE NOT NULL,
    category_id VARCHAR(50) NOT NULL REFERENCES orchestration_pattern_categories(category_id

    -- Pattern metadata
    name VARCHAR(200) NOT NULL,
    description TEXT NOT NULL,
    research_basis TEXT,                          -- Academic source if applicable

    -- Implementation details
    complexity VARCHAR(20) NOT NULL CHECK (complexity IN ('low', 'medium', 'high', 'very_hig
    impact VARCHAR(20) NOT NULL CHECK (impact IN ('low', 'medium', 'high', 'transformative')
    implemented_by TEXT[],                        -- Competitors who implement this
    implementation_status VARCHAR(30) DEFAULT 'planned',

    -- Semantic embedding for Neural Engine matching
    semantic_embedding VECTOR(768),

    -- Execution characteristics
    execution_type VARCHAR(20) DEFAULT 'serial' CHECK (execution_type IN ('serial', 'paralle
    parallelizable BOOLEAN DEFAULT FALSE,
    typical_model_count INTEGER DEFAULT 1,
    typical_latency_ms INTEGER,
    typical_cost_multiplier DECIMAL(4,2) DEFAULT 1.0,

    -- Pattern definition (DAG structure)
    pattern_definition JSONB NOT NULL,

    -- Trigger conditions
    trigger_keywords TEXT[],
    trigger_intents TEXT[],
    trigger_complexity_range NUMRANGE,

    -- Requirements
    required_capabilities TEXT[],
    min_tier INTEGER DEFAULT 1,

    -- Stats
    usage_count INTEGER DEFAULT 0,
    avg_satisfaction_score DECIMAL(3,2),

    enabled BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
```

```
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Insert orchestration pattern categories (10 categories)
INSERT INTO orchestration_pattern_categories (category_id, name, description, display_order,
('multi_model', 'Multi-Model Coordination', 'Patterns for coordinating multiple AI models',
('sequential', 'Sequential & Pipeline', 'Step-by-step processing patterns', 2, 15),
('verification', 'Verification & Fact-Checking', 'Patterns for validating AI outputs', 3, 12
('debate', 'Debate & Adversarial Review', 'Patterns for adversarial evaluation', 4, 12),
('reasoning', 'Reasoning Enhancement', 'Patterns for improving reasoning quality', 5, 15),
('agent', 'Agent Architectures', 'Multi-agent coordination patterns', 6, 18),
('tool', 'Tool Use & Integration', 'Patterns for external tool integration', 7, 10),
('memory', 'Memory & Personalization', 'Patterns for context and personalization', 8, 8),
('user_facing', 'User-Facing Workflow Features', 'Patterns for user interaction', 9, 12),
('emerging', 'Emerging & Cutting-Edge', 'Experimental and research-stage patterns', 10, 15)
ON CONFLICT (category_id) DO UPDATE SET pattern_count = EXCLUDED.pattern_count;

-- ============================================================================
-- PART 2: THINK TANK PRODUCTION WORKFLOWS (127 workflows)
-- ============================================================================

CREATE TABLE IF NOT EXISTS production_workflow_categories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    category_id VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    display_order INTEGER DEFAULT 0,
    workflow_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS production_workflows (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    workflow_id VARCHAR(100) UNIQUE NOT NULL,
    category_id VARCHAR(50) NOT NULL REFERENCES production_workflow_categories(category_id),

    name VARCHAR(200) NOT NULL,
    description TEXT NOT NULL,
    primary_deliverable VARCHAR(200),
    semantic_embedding VECTOR(768),
    workflow_definition JSONB NOT NULL,
    input_schema JSONB,
    output_schema JSONB,
    complexity VARCHAR(20) DEFAULT 'medium',
    typical_duration_minutes INTEGER,
```

```
    trigger_keywords TEXT[],
    trigger_domains TEXT[],
    required_patterns TEXT[],
    required_capabilities TEXT[],
    min_tier INTEGER DEFAULT 1,

    usage_count INTEGER DEFAULT 0,
    avg_satisfaction_score DECIMAL(3,2),
    enabled BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Insert production workflow categories (12 categories)
INSERT INTO production_workflow_categories (category_id, name, description, display_order, w
('technical', 'Technical Documentation', 'Engineering and technical writing', 1, 14),
('research', 'Research & Analysis', 'Research papers and analysis reports', 2, 12),
('business', 'Business Documents', 'Business plans, proposals, reports', 3, 10),
('legal', 'Legal Documents', 'Contracts, policies, compliance', 4, 10),
('medical', 'Medical & Healthcare', 'Medical documentation and protocols', 5, 10),
('education', 'Education & Training', 'Learning materials and curricula', 6, 10),
('marketing', 'Marketing & Communications', 'Marketing content and campaigns', 7, 10),
('financial', 'Financial Documents', 'Financial reports and analysis', 8, 10),
('creative', 'Creative Production', 'Creative works and design assets', 9, 10),
('trades', 'Skilled Trades', 'Trade-specific documentation', 10, 10),
('scientific', 'Scientific Workflows', 'Scientific analysis and research', 11, 8),
('software', 'Software Development', 'Software documentation and specs', 12, 13)
ON CONFLICT (category_id) DO UPDATE SET workflow_count = EXCLUDED.workflow_count;

-- ============================================================================
-- PART 3: THINK TANK DOMAIN REGISTRY (834 domains)
-- ============================================================================

CREATE TABLE IF NOT EXISTS domain_categories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    category_id VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    display_order INTEGER DEFAULT 0,
    domain_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS specialized_domains (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    domain_id VARCHAR(100) UNIQUE NOT NULL,
```

```
    category_id VARCHAR(50) NOT NULL REFERENCES domain_categories(category_id),
    parent_domain_id VARCHAR(100) REFERENCES specialized_domains(domain_id),

    name VARCHAR(200) NOT NULL,
    description TEXT,
    semantic_embedding VECTOR(768),

    expert_context TEXT,
    terminology JSONB,
    standards TEXT[],
    best_practices TEXT[],

    keywords TEXT[],
    related_domains TEXT[],
    preferred_models TEXT[],
    preferred_patterns TEXT[],
    preferred_workflows TEXT[],

    usage_count INTEGER DEFAULT 0,
    enabled BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Insert domain categories (17 categories, 834 domains total)
INSERT INTO domain_categories (category_id, name, description, display_order, domain_count)
('sciences', 'Sciences', 'Physics, Chemistry, Biology, Earth Sciences, Materials', 1, 56),
('mathematics', 'Mathematics & Logic', 'Algebra, Calculus, Statistics, Logic', 2, 32),
('computer_science', 'Computer Science & Programming', 'Languages, Development, Theory', 3,
('ai', 'Artificial Intelligence', 'ML, NLP, Computer Vision, AI Systems', 4, 48),
('engineering', 'Engineering', 'Mechanical, Electrical, Civil, Chemical', 5, 72),
('medicine', 'Medicine & Healthcare', 'Clinical, Surgery, Allied Health', 6, 89),
('mental_health', 'Mental Health & Psychology', 'Clinical, Counseling, Therapy', 7, 34),
('fitness', 'Fitness, Therapy & Nutrition', 'Training, Nutrition, Wellness', 8, 42),
('humanities', 'Humanities', 'Philosophy, History, Literature, Arts', 9, 68),
('social_sciences', 'Social Sciences', 'Sociology, Economics, Political Science', 10, 41),
('business', 'Business & Management', 'Strategy, Finance, Marketing, Operations', 11, 58),
('legal', 'Law & Legal', 'Corporate, IP, Employment, Litigation', 12, 34),
('education', 'Education', 'Curriculum, Assessment, Special Ed', 13, 29),
('trades', 'Skilled Trades', 'Construction, Mechanical, Electrical, Automotive', 14, 44),
('arts_design', 'Arts, Design & Creativity', 'Visual, Performing, Digital', 15, 52),
('communication', 'Communication & Media', 'Journalism, PR, Social Media', 16, 26),
('emerging', 'Interdisciplinary & Emerging', 'Sustainability, AI Ethics, Futures', 17, 31)
ON CONFLICT (category_id) DO UPDATE SET domain_count = EXCLUDED.domain_count;

-- ============================================================================
-- PART 4: PER-USER NEURAL MODELS
```

```
-- ============================================================================

CREATE TABLE IF NOT EXISTS user_neural_models (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Profile embeddings (768-dim vectors)
    profile_embedding VECTOR(768),
    query_style_embedding VECTOR(768),
    feedback_pattern_embedding VECTOR(768),

    -- Structured preferences
    model_preferences JSONB DEFAULT '{}',
    workflow_preferences JSONB DEFAULT '{
        "prefersVerification": 0.5,
        "toleratesLatency": 0.5,
        "costSensitivity": 0.5,
        "prefersExplanation": 0.5,
        "prefersDetailedResponses": 0.5,
        "prefersStructuredOutput": 0.5
    }',
    quality_thresholds JSONB DEFAULT '{
        "minimumAcceptableQuality": 0.6,
        "qualityVsSpeedTradeoff": 0.5,
        "qualityVsCostTradeoff": 0.5
    }',

    -- Behavioral patterns
    behavioral_patterns JSONB DEFAULT '{
        "typicalQueryLength": "medium",
        "typicalComplexity": "moderate",
        "frequentDomains": [],
        "frequentWorkflows": [],
        "peakUsageHours": [],
        "feedbackFrequency": 0.5
    }',

    -- Confidence and training stats
    overall_confidence DECIMAL(5,4) DEFAULT 0,
    total_interactions INTEGER DEFAULT 0,
    total_feedback_events INTEGER DEFAULT 0,
    training_sample_count INTEGER DEFAULT 0,

    -- Admin overrides
    admin_overrides JSONB DEFAULT '{
```

```
        "forceWorkflow": null,
        "forceModel": null,
        "disablePersonalization": false,
        "customParameters": {}
    }',

    last_training_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(user_id, tenant_id)
);

-- Domain-specific embeddings per user
CREATE TABLE IF NOT EXISTS user_domain_embeddings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_model_id UUID NOT NULL REFERENCES user_neural_models(id) ON DELETE CASCADE,
    domain_id VARCHAR(100) NOT NULL,
    embedding VECTOR(768),
    confidence DECIMAL(5,4) DEFAULT 0,
    sample_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    UNIQUE(user_model_id, domain_id)
);

-- ============================================================================
-- PART 5: NEURAL ENGINE CONFIGURATION (Admin Editable)
-- ============================================================================

CREATE TABLE IF NOT EXISTS neural_engine_config (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id) ON DELETE CASCADE,

    config JSONB NOT NULL DEFAULT '{
        "learning": {
            "learningRate": 0.01,
            "userModelUpdateFrequency": "realtime",
            "minSamplesForPersonalization": 20,
            "confidenceDecayRate": 0.001
        },
        "userModel": {
            "embeddingDimension": 768,
            "minConfidenceThreshold": 0.6,
            "coldStartStrategy": "global_defaults"
        },
```

```
        "construction": {
            "maxNodesPerWorkflow": 20,
            "preferTemplates": true,
            "templateMatchThreshold": 0.8
        },
        "monitoring": {
            "realTimeSteeringEnabled": true,
            "anomalyDetectionSensitivity": 0.7,
            "maxSteeringActionsPerExecution": 3
        },
        "modelSelection": {
            "neuralWeightInScoring": 0.6,
            "fallbackToDefaults": true
        },
        "scope": {
            "enableGlobalLearning": true,
            "enableTenantLearning": true,
            "enableUserLearning": true,
            "globalLearningWeight": 0.2,
            "tenantLearningWeight": 0.3,
            "userLearningWeight": 0.5
        }
    }',

    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tenant_id)
);

-- ============================================================================
-- PART 6: BRAIN GOVERNOR CONFIGURATION (Admin Editable)
-- ============================================================================

CREATE TABLE IF NOT EXISTS brain_config (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id) ON DELETE CASCADE,

    config JSONB NOT NULL DEFAULT '{
        "costs": {
            "maxCostPerRequest": 0.50,
            "maxCostPerHour": 10.00,
            "maxCostPerDay": 100.00,
            "costAlertThreshold": 0.8
        },
```

```
        "latency": {
            "maxLatencyMs": 30000,
            "latencyWarningThreshold": 0.7
        },
        "quality": {
            "minimumConfidenceThreshold": 0.5,
            "requireVerificationAboveComplexity": 0.8,
            "maxRetriesPerNode": 3
        },
        "neuralTrust": {
            "trustThreshold": 0.7,
            "autoApproveKnownWorkflows": true,
            "requireHumanApprovalBelow": 0.3
        },
        "steering": {
            "allowModelSwitching": true,
            "allowNodeSkipping": false,
            "allowWorkflowModification": true
        },
        "compliance": {
            "requireHIPAAForMedical": true,
            "requireAuditLogging": true,
            "piiDetectionEnabled": true
        }
    }',

    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tenant_id)
);

-- Brain policies and decision audit
CREATE TABLE IF NOT EXISTS brain_policies (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    name VARCHAR(200) NOT NULL,
    description TEXT,
    policy_type VARCHAR(50) NOT NULL,
    conditions JSONB NOT NULL DEFAULT '[]',
    actions JSONB NOT NULL DEFAULT '[]',
    enabled BOOLEAN DEFAULT TRUE,
    priority INTEGER DEFAULT 0,
    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW()
```

```
);

CREATE TABLE IF NOT EXISTS brain_decisions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    execution_id UUID NOT NULL,
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    session_id UUID REFERENCES concurrent_sessions(id),
    decision_type VARCHAR(50) NOT NULL,
    proposal JSONB NOT NULL,
    decision JSONB NOT NULL,
    approved BOOLEAN NOT NULL,
    modifications JSONB,
    guardrails_applied JSONB,
    rejection_reason TEXT,
    decided_by VARCHAR(50) NOT NULL,
    admin_id UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- ============================================================================
-- PART 7: CONSTRUCTED WORKFLOWS & NEURAL EVENTS
-- ============================================================================

CREATE TABLE IF NOT EXISTS constructed_workflows (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    name VARCHAR(200),
    description TEXT,
    nodes JSONB NOT NULL,
    edges JSONB NOT NULL,
    generation_method VARCHAR(50) NOT NULL,
    source_template_id UUID REFERENCES workflow_definitions(id),
    source_pattern_ids UUID[],
    auto_metadata JSONB DEFAULT '{}',
    admin_metadata_overrides JSONB DEFAULT '{}',
    reasoning JSONB DEFAULT '{}',
    alternatives JSONB DEFAULT '[]',
    brain_approved BOOLEAN DEFAULT TRUE,
    brain_modifications JSONB,
    usage_count INTEGER DEFAULT 0,
    avg_satisfaction_score DECIMAL(3,2),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS neural_execution_events (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    execution_id UUID NOT NULL,
    node_id VARCHAR(100),
    session_id UUID REFERENCES concurrent_sessions(id),
    event_type VARCHAR(50) NOT NULL,
    event_data JSONB NOT NULL,
    brain_notified BOOLEAN DEFAULT FALSE,
    brain_response JSONB,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS neural_learning_signals (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    signal_type VARCHAR(50) NOT NULL,
    user_id UUID REFERENCES users(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    workflow_id UUID,
    model_id VARCHAR(100),
    node_id VARCHAR(100),
    session_id UUID REFERENCES concurrent_sessions(id),
    signal_value DECIMAL(5,4) NOT NULL,
    confidence DECIMAL(5,4) DEFAULT 1.0,
    weight DECIMAL(5,4) DEFAULT 1.0,
    source VARCHAR(50) NOT NULL,
    processed BOOLEAN DEFAULT FALSE,
    processed_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- ============================================================================
-- PART 8: INDEXES
-- ============================================================================

CREATE INDEX IF NOT EXISTS idx_orchestration_patterns_category ON orchestration_patterns(cat
CREATE INDEX IF NOT EXISTS idx_orchestration_patterns_enabled ON orchestration_patterns(enab
CREATE INDEX IF NOT EXISTS idx_production_workflows_category ON production_workflows(categor
CREATE INDEX IF NOT EXISTS idx_specialized_domains_category ON specialized_domains(category_
CREATE INDEX IF NOT EXISTS idx_user_neural_models_user ON user_neural_models(user_id);
CREATE INDEX IF NOT EXISTS idx_user_neural_models_tenant ON user_neural_models(tenant_id);

-- Vector similarity indexes
CREATE INDEX IF NOT EXISTS idx_orchestration_patterns_embedding ON orchestration_patterns
    USING ivfflat (semantic_embedding vector_cosine_ops) WITH (lists = 50);
CREATE INDEX IF NOT EXISTS idx_production_workflows_embedding ON production_workflows
    USING ivfflat (semantic_embedding vector_cosine_ops) WITH (lists = 50);
CREATE INDEX IF NOT EXISTS idx_specialized_domains_embedding ON specialized_domains
```

```
    USING ivfflat (semantic_embedding vector_cosine_ops) WITH (lists = 100);

-- Neural event indexes for concurrent execution
CREATE INDEX IF NOT EXISTS idx_neural_execution_events_session ON neural_execution_events(se
CREATE INDEX IF NOT EXISTS idx_neural_learning_signals_session ON neural_learning_signals(se
CREATE INDEX IF NOT EXISTS idx_brain_decisions_session ON brain_decisions(session_id);

-- ============================================================================
-- PART 9: ROW LEVEL SECURITY
-- ============================================================================

ALTER TABLE user_neural_models ENABLE ROW LEVEL SECURITY;
ALTER TABLE brain_policies ENABLE ROW LEVEL SECURITY;
ALTER TABLE brain_decisions ENABLE ROW LEVEL SECURITY;
ALTER TABLE constructed_workflows ENABLE ROW LEVEL SECURITY;

CREATE POLICY user_neural_models_isolation ON user_neural_models
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY brain_policies_isolation ON brain_policies
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY brain_decisions_isolation ON brain_decisions
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY constructed_workflows_isolation ON constructed_workflows
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

-- ============================================================================
-- PART 10: HELPER FUNCTIONS
-- ============================================================================

-- Function: Get user's neural model with fallbacks
CREATE OR REPLACE FUNCTION get_user_neural_model(p_user_id UUID, p_tenant_id UUID)
RETURNS JSONB AS $$
DECLARE
    user_model JSONB;
BEGIN
    SELECT jsonb_build_object(
        'userId', user_id,
        'modelPreferences', model_preferences,
        'workflowPreferences', workflow_preferences,
        'qualityThresholds', quality_thresholds,
        'behavioralPatterns', behavioral_patterns,
        'overallConfidence', overall_confidence,
        'adminOverrides', admin_overrides
    ) INTO user_model
    FROM user_neural_models
    WHERE user_id = p_user_id AND tenant_id = p_tenant_id;
```

```
    RETURN COALESCE(user_model, '{}');
END;
$$ LANGUAGE plpgsql;

-- Function: Record learning signal (atomic for concurrent execution)
CREATE OR REPLACE FUNCTION record_learning_signal(
    p_signal_type VARCHAR(50),
    p_user_id UUID,
    p_tenant_id UUID,
    p_workflow_id UUID,
    p_model_id VARCHAR(100),
    p_session_id UUID,
    p_signal_value DECIMAL,
    p_source VARCHAR(50)
)
RETURNS UUID AS $$
DECLARE
    signal_id UUID;
BEGIN
    INSERT INTO neural_learning_signals (
        signal_type, user_id, tenant_id, workflow_id, model_id,
        session_id, signal_value, source
    ) VALUES (
        p_signal_type, p_user_id, p_tenant_id, p_workflow_id, p_model_id,
        p_session_id, p_signal_value, p_source
    ) RETURNING id INTO signal_id;

    RETURN signal_id;
END;
$$ LANGUAGE plpgsql;

-- Add session_id to execution_manifests for concurrent tracking
DO $$
BEGIN
    IF NOT EXISTS (
        SELECT 1 FROM information_schema.columns
        WHERE table_name = 'execution_manifests' AND column_name = 'session_id'
    ) THEN
        ALTER TABLE execution_manifests ADD COLUMN session_id UUID REFERENCES concurrent_ses
        CREATE INDEX idx_execution_manifests_session ON execution_manifests(session_id);
    END IF;
END $$;

-- Statistics view
CREATE OR REPLACE VIEW neural_orchestration_stats AS
```

```sql
SELECT
    (SELECT COUNT(*) FROM orchestration_patterns WHERE enabled = TRUE) as total_patterns,
    (SELECT COUNT(*) FROM orchestration_patterns WHERE implementation_status = 'implemented'
    (SELECT COUNT(*) FROM production_workflows WHERE enabled = TRUE) as total_workflows,
    (SELECT COUNT(*) FROM specialized_domains WHERE enabled = TRUE) as total_domains,
    (SELECT COUNT(*) FROM user_neural_models) as total_user_models,
    (SELECT AVG(overall_confidence) FROM user_neural_models) as avg_user_confidence,
    (SELECT COUNT(*) FROM constructed_workflows) as total_constructed_workflows,
    (SELECT COUNT(*) FROM neural_execution_events WHERE created_at > NOW() - INTERVAL '24 ho
```

38.4 Seed Data: Sample Orchestration Patterns

migrations/038b_seed_orchestration_patterns.sql

```sql
-- Sample of 127 Orchestration Patterns from Think Tank Compendium
-- Full seed data includes all 127 patterns

-- Multi-Model Coordination Patterns (10)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity,
VALUES
('side_by_side_comparison', 'multi_model', 'Side-by-Side Comparison', 'Send identical prompt
('consensus_voting', 'multi_model', 'Consensus Voting', 'Query multiple models, aggregate vi
('ensemble_synthesis', 'multi_model', 'Ensemble Response Synthesis', 'Combine responses fro
('intelligent_routing', 'multi_model', 'Intelligent Model Routing', 'Auto-select optimal mo
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Sequential Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity,
VALUES
('draft_critique_revise', 'sequential', 'Draft → Critique → Revise', 'Generate, evaluate, i
('research_analyze_report', 'sequential', 'Research → Analyze → Report', 'Sequential researc
('plan_execute_verify', 'sequential', 'Plan → Execute → Verify', 'Agentic task completion pa
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Verification Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity,
VALUES
('red_team_attack', 'verification', 'Red Team Attack', 'Adversarial model challenges primary
('chain_of_verification', 'verification', 'Chain of Verification (CoVe)', 'Generate claims -
('hallucination_detection', 'verification', 'Hallucination Detection', 'Identify unsupporte
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Debate Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity,
VALUES
```

```
('ai_debate', 'debate', 'AI Debate', 'Two models argue opposing positions', 'medium', 'high'
('round_table_consensus', 'debate', 'Round Table Consensus', 'Multiple agents discuss to rea
('ai_judge', 'debate', 'AI Judge', 'Third model evaluates debate outcome', 'medium', 'high',
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Reasoning Enhancement Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity,
VALUES
('chain_of_thought', 'reasoning', 'Chain of Thought (CoT)', 'Step-by-step reasoning', 'low',
('tree_of_thoughts', 'reasoning', 'Tree of Thoughts (ToT)', 'Explore multiple reasoning bran
('self_consistency', 'reasoning', 'Self-Consistency', 'Sample multiple CoT paths, vote on an
('self_refine', 'reasoning', 'Self-Refine Loop', 'Iterative self-improvement', 'medium', 'hi
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Note: Full implementation includes all 127 patterns from Think Tank Compendium
```

38.5 API Endpoints Summary

Neural Orchestration Admin Endpoints

Method	Endpoint	Description
GET	`/api/v2/admin/neural/config`	Get Neural Engine config
PUT	`/api/v2/admin/neural/config`	Update Neural Engine config
GET	`/api/v2/admin/brain/config`	Get Brain Governor config
PUT	`/api/v2/admin/brain/config`	Update Brain Governor config
GET	`/api/v2/admin/patterns`	List orchestration patterns
PUT	`/api/v2/admin/patterns/{pattern_id}`	Update pattern status
GET	`/api/v2/admin/workflows/production`	List production workflows
GET	`/api/v2/admin/domains`	List specialized domains
GET	`/api/v2/admin/user-models`	List user neural models
PUT	`/api/v2/admin/user-models/{user_id}/overrides`	Set admin overrides
DELETE	`/api/v2/admin/user-models/{user_id}`	Reset user model
GET	`/api/v2/admin/registry/stats`	Get registry statistics

Workflow Editor Endpoints

Method	Endpoint	Description
GET	`/api/v2/admin/workflows/templates`	List workflow templates
POST	`/api/v2/admin/workflows/templates`	Create template
PUT	`/api/v2/admin/workflows/templates/{id}`	Update template
DELETE	`/api/v2/admin/workflows/templates/{id}`	Delete template
POST	`/api/v2/admin/workflows/generate-metadata`	AI generate metadata

Method	Endpoint	Description
POST	`/api/v2/admin/workflows/test`	Test workflow execution

Client Decision Transparency Endpoints

Method	Endpoint	Description
GET	`/api/v2/decision/{execution_id}`	Get decision transparency
POST	`/api/v2/decision/{execution_id}/override`	Submit override request

38.6 Swift Deployer v2 Updates

The Swift Deployment App now includes:

1. **Neural Engine Configuration View**
   - Learning parameters (rate, frequency, decay)
   - User model settings (cold start, confidence thresholds)
   - Construction parameters (max nodes, template preferences)
   - Real-time monitoring toggles
2. **Brain Governor Configuration View**
   - Cost controls (per-request, per-hour, per-day limits)
   - Latency controls (max latency, warning thresholds)
   - Quality controls (confidence, verification requirements)
   - Neural trust settings (approval thresholds)
   - Steering controls (model switching, node skipping)
   - Compliance settings (HIPAA, audit logging)
3. **Workflow Registry Browser**
   - View 127 orchestration patterns by category
   - View 127 production workflows by category
   - View 834 specialized domains by category
   - Search and filter capabilities
   - Usage statistics dashboard

38.7 Verification & Deployment

Pre-Deployment Checklist

```
# 1. Apply database migrations
psql $DATABASE_URL -f migrations/038_neural_orchestration_registry.sql
psql $DATABASE_URL -f migrations/038b_seed_orchestration_patterns.sql

# 2. Verify tables created
```

```
psql $DATABASE_URL -c "SELECT * FROM neural_orchestration_stats"

# 3. Verify registry counts
psql $DATABASE_URL -c "SELECT COUNT(*) as patterns FROM orchestration_patterns"
psql $DATABASE_URL -c "SELECT COUNT(*) as categories FROM domain_categories"

# 4. Test Neural Engine config
curl https://api.YOUR_DOMAIN.com/api/v2/admin/neural/config \
  -H "Authorization: Bearer $ADMIN_TOKEN"

# 5. Test decision transparency
curl https://api.YOUR_DOMAIN.com/api/v2/decision/{executionId} \
  -H "Authorization: Bearer $TOKEN"
```

Summary v4.4.0

RADIANT v4.4.0 (PROMPT-18) adds **Neural-First Orchestration & Think Tank Workflow Registry**:

Section 38: Neural-First Orchestration (v4.4.0)

1. **Think Tank Workflow Registry** (Database Schema)
   - 127 orchestration patterns across 10 categories
   - 127 production workflows across 12 categories
   - 834 specialized domains across 17 categories
   - Full semantic embeddings for Neural Engine matching
2. **Neural Engine Enhancements**
   - Per-user neural models with 768-dim embeddings
   - Orchestration constructor (dynamic workflow generation)
   - Real-time steering during execution
   - Learning from concurrent sessions (atomic updates)
3. **Brain Governor Enhancements**
   - Full admin-editable configuration
   - Policy engine with conditions and actions
   - Decision audit logging
   - Concurrent session awareness
4. **Visual Workflow Editor**
   - Drag-and-drop node placement
   - 12 node type categories
   - Neural I/O connector visualization
   - Auto-generated metadata
5. **Client Decision Transparency**
   - Reasoning exposed to Think Tank
   - Confidence scores with explanations

- Alternatives with tradeoffs
- Override options with impact estimation

6. **Swift Deployer v2**
   - Neural Engine configuration UI
   - Brain Governor configuration UI
   - Workflow registry browser
7. **Concurrent Execution Support**
   - Session-aware billing aggregation
   - Session-tagged feedback
   - Atomic neural model updates

Design Philosophy (v4.4.0)

- **Neural-First** - Neural Engine is the fabric (60% weight), Brain is the governor
- **Dynamic Construction** - Workflows built from primitives, not just template selection
- **Per-User Learning** - Personalized embeddings and preferences
- **Real-Time Steering** - Adjustments during execution, not just post-hoc
- **Full Transparency** - Clients see reasoning, confidence, alternatives
- **Admin Control** - All parameters editable through dashboard
- **Concurrent Aware** - Proper handling of parallel user sessions

Also includes all v4.3.0 features:

- Feedback Learning System
- Neural Engine Loop
- Multi-Language Voice Feedback
- Implicit Signals
- A/B Testing Framework

Total Sections: 40 (0-39) Total Lines: ~53,000 Total Size: ~2.3MB

SECTION-39-WORKFLOW-PROPOSALS

SECTION 39: DYNAMIC WORKFLOW PROPOSAL SYSTEM (v4.5.0)

This section implements the Dynamic Workflow Proposal System for v4.5.0

##
39.1
Overview

The Dynamic Workflow Proposal System enables the Brain and Neural Engine to collaboratively identify user needs that aren't well-served by existing workflows, propose new workflows to solve these problems, and submit them for administ-

---

Key Design Principles

1. **Evidence-Based Proposals** - Only propose when sufficient user need is demonstrated
2. **Threshold Gating** - Multiple thresholds must be met before proposal generation
3. **Brain Governor Oversight** - Brain reviews all proposals be-

Evidence Types

Evidence Type	Description	Weight
`workflow_failure`	Existing workflow failed to complete	0.40
`negative_feedback`	User gave thumbs down or negative rating	0.35
`manual_override`	User manually switched model/approach	

39.2 Database Schema

```sql
-- ============================================================================
-- SECTION 39: DYNAMIC WORKFLOW PROPOSAL SYSTEM
-- Migration: 039_dynamic_workflow_proposals.sql
-- Version: 4.5.0
-- ============================================================================

-- ============================================================================
-- 39.2.1 Need Pattern Detection Tables
-- ============================================================================

-- Track emerging user need patterns that could justify new workflows
CREATE TABLE neural_need_patterns (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Pattern identification
    pattern_hash VARCHAR(64) NOT NULL,  -- SHA-256 of normalized pattern signature
    pattern_signature JSONB NOT NULL,   -- Structured pattern definition
    pattern_embedding VECTOR(768),      -- Neural embedding for similarity search

    -- Pattern metadata
    pattern_name VARCHAR(255) NOT NULL,
    pattern_description TEXT,
    detected_intent TEXT NOT NULL,      -- What the user was trying to accomplish
    existing_workflow_gaps TEXT[],      -- Which existing workflows fail this case

    -- Evidence accumulation
    total_evidence_score DECIMAL(10,4) DEFAULT 0,
    evidence_count INTEGER DEFAULT 0,
    unique_users_affected INTEGER DEFAULT 0,
    first_occurrence TIMESTAMPTZ DEFAULT NOW(),
    last_occurrence TIMESTAMPTZ DEFAULT NOW(),

    -- Threshold tracking
    occurrence_threshold_met BOOLEAN DEFAULT FALSE,
    impact_threshold_met BOOLEAN DEFAULT FALSE,
    confidence_threshold_met BOOLEAN DEFAULT FALSE,

    -- Status tracking
    status VARCHAR(50) DEFAULT 'accumulating',  -- accumulating, threshold_met, proposal_ge
    proposal_id UUID,  -- Link to generated proposal if threshold met

    -- Audit
    created_at TIMESTAMPTZ DEFAULT NOW(),
```

```
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    CONSTRAINT unique_pattern_per_tenant UNIQUE(tenant_id, pattern_hash)
);

-- Individual evidence records contributing to need patterns
CREATE TABLE neural_need_evidence (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    pattern_id UUID NOT NULL REFERENCES neural_need_patterns(id) ON DELETE CASCADE,

    -- Evidence source
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    session_id UUID,
    execution_id UUID,  -- Link to manifest for full context

    -- Evidence details
    evidence_type VARCHAR(50) NOT NULL,  -- workflow_failure, negative_feedback, manual_ove
    evidence_weight DECIMAL(5,4) NOT NULL,
    evidence_data JSONB NOT NULL,  -- Detailed context

    -- User intent capture
    original_request TEXT,           -- What user asked for
    attempted_workflow_id UUID,      -- Which workflow was tried
    failure_reason TEXT,             -- Why it failed or underperformed
    user_feedback TEXT,              -- Any explicit user feedback

    -- Temporal
    occurred_at TIMESTAMPTZ DEFAULT NOW(),

    -- Audit
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- ============================================================================
-- 39.2.2 Proposal Tables
-- ============================================================================

-- Workflow proposals generated by Neural Engine
CREATE TABLE workflow_proposals (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Proposal identification
    proposal_code VARCHAR(50) NOT NULL,  -- e.g., WP-2024-001
    proposal_name VARCHAR(255) NOT NULL,
```

```
proposal_description TEXT NOT NULL,

-- Source pattern
source_pattern_id UUID NOT NULL REFERENCES neural_need_patterns(id),

-- Proposed workflow structure
proposed_workflow JSONB NOT NULL,  -- Full workflow DAG definition
workflow_category VARCHAR(100),
workflow_type VARCHAR(50),  -- orchestration_pattern, production_workflow, hybrid
estimated_complexity VARCHAR(20),  -- simple, moderate, complex, advanced

-- Neural Engine analysis
neural_confidence DECIMAL(5,4) NOT NULL,  -- How confident Neural is this solves the nee
neural_reasoning TEXT NOT NULL,           -- Why Neural proposed this structure
estimated_quality_improvement DECIMAL(5,4),
estimated_coverage_percentage DECIMAL(5,4),  -- % of evidence cases this would solve

-- Brain Governor review
brain_approved BOOLEAN,
brain_review_timestamp TIMESTAMPTZ,
brain_risk_assessment JSONB,  -- cost_risk, latency_risk, quality_risk, compliance_risk
brain_veto_reason TEXT,       -- If vetoed, why
brain_modifications JSONB,    -- Any modifications Brain suggested

-- Evidence summary
evidence_count INTEGER NOT NULL,
unique_users_affected INTEGER NOT NULL,
total_evidence_score DECIMAL(10,4) NOT NULL,
evidence_time_span_days INTEGER NOT NULL,
evidence_summary JSONB NOT NULL,  -- Aggregated evidence statistics

-- Admin review
admin_status VARCHAR(50) DEFAULT 'pending_brain',  -- pending_brain, pending_admin, appr
admin_reviewer_id UUID REFERENCES users(id),
admin_review_timestamp TIMESTAMPTZ,
admin_notes TEXT,
admin_modifications JSONB,  -- Any modifications admin made

-- Testing
test_status VARCHAR(50),  -- not_tested, testing, passed, failed
test_results JSONB,
test_execution_ids UUID[],

-- Publishing
published_workflow_id UUID,  -- Link to published workflow if approved
published_at TIMESTAMPTZ,
```

```
    -- Rate limiting
    proposal_batch_id UUID,  -- Group proposals from same analysis run

    -- Audit
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    CONSTRAINT unique_proposal_code UNIQUE(tenant_id, proposal_code)
);

-- Proposal review history for audit trail
CREATE TABLE workflow_proposal_reviews (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    proposal_id UUID NOT NULL REFERENCES workflow_proposals(id) ON DELETE CASCADE,

    -- Reviewer
    reviewer_type VARCHAR(20) NOT NULL,  -- brain, admin
    reviewer_id UUID,  -- NULL for brain, user_id for admin

    -- Review details
    action VARCHAR(50) NOT NULL,  -- approve, decline, modify, request_test, escalate
    previous_status VARCHAR(50),
    new_status VARCHAR(50),

    -- Reasoning
    review_notes TEXT,
    modifications_made JSONB,
    risk_assessment JSONB,

    -- Audit
    reviewed_at TIMESTAMPTZ DEFAULT NOW()
);

-- ============================================================================
-- 39.2.3 Threshold Configuration Tables
-- ============================================================================

-- Configurable thresholds for proposal generation
CREATE TABLE proposal_threshold_config (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Occurrence thresholds
    min_evidence_count INTEGER DEFAULT 5,           -- Minimum evidence records needed
    min_unique_users INTEGER DEFAULT 3,             -- Minimum unique users affected
```

```
    min_time_span_hours INTEGER DEFAULT 24,          -- Minimum time span of evidence
    max_time_span_days INTEGER DEFAULT 30,           -- Maximum lookback for evidence

    -- Impact thresholds
    min_total_evidence_score DECIMAL(5,4) DEFAULT 0.60,  -- Minimum cumulative evidence sco
    min_avg_evidence_weight DECIMAL(5,4) DEFAULT 0.20,   -- Minimum average evidence weight

    -- Neural confidence thresholds
    min_neural_confidence DECIMAL(5,4) DEFAULT 0.75,     -- Minimum Neural confidence to pr
    min_coverage_estimate DECIMAL(5,4) DEFAULT 0.60,     -- Minimum estimated coverage

    -- Brain approval thresholds
    max_cost_risk DECIMAL(5,4) DEFAULT 0.30,             -- Maximum acceptable cost risk
    max_latency_risk DECIMAL(5,4) DEFAULT 0.40,          -- Maximum acceptable latency risk
    min_quality_confidence DECIMAL(5,4) DEFAULT 0.70,    -- Minimum quality confidence
    max_compliance_risk DECIMAL(5,4) DEFAULT 0.10,       -- Maximum compliance risk

    -- Rate limiting
    max_proposals_per_day INTEGER DEFAULT 10,
    max_proposals_per_week INTEGER DEFAULT 30,
    cooldown_after_decline_hours INTEGER DEFAULT 72,     -- Wait time after admin decline

    -- Auto-approve settings (disabled by default)
    auto_approve_enabled BOOLEAN DEFAULT FALSE,
    auto_approve_min_confidence DECIMAL(5,4) DEFAULT 0.95,
    auto_approve_max_complexity VARCHAR(20) DEFAULT 'simple',

    -- Audit
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    updated_by UUID REFERENCES users(id),

    CONSTRAINT one_config_per_tenant UNIQUE(tenant_id)
);

-- Evidence type weight configuration (admin-editable)
CREATE TABLE evidence_weight_config (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    evidence_type VARCHAR(50) NOT NULL,
    weight DECIMAL(5,4) NOT NULL,
    description TEXT,
    enabled BOOLEAN DEFAULT TRUE,

    -- Audit
```

```
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    updated_by UUID REFERENCES users(id),

    CONSTRAINT unique_evidence_type_per_tenant UNIQUE(tenant_id, evidence_type)
);

-- ============================================================================
-- 39.2.4 Neural Engine Learning Persistence
-- ============================================================================

-- Ensure Neural Engine data is persisted (extends Section 38 if not present)
CREATE TABLE IF NOT EXISTS neural_learning_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Session identification
    session_type VARCHAR(50) NOT NULL,  -- pattern_detection, proposal_generation, learning_
    started_at TIMESTAMPTZ DEFAULT NOW(),
    completed_at TIMESTAMPTZ,

    -- Learning data
    patterns_analyzed INTEGER DEFAULT 0,
    patterns_updated INTEGER DEFAULT 0,
    proposals_generated INTEGER DEFAULT 0,
    evidence_processed INTEGER DEFAULT 0,

    -- Model state snapshots
    model_state_before JSONB,
    model_state_after JSONB,

    -- Performance
    processing_time_ms INTEGER,
    memory_used_mb INTEGER,

    -- Audit
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Neural embedding updates log
CREATE TABLE neural_embedding_updates (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Source
    entity_type VARCHAR(50) NOT NULL,  -- pattern, user_model, workflow
```

```
    entity_id UUID NOT NULL,

    -- Embedding change
    previous_embedding VECTOR(768),
    new_embedding VECTOR(768),
    embedding_delta_magnitude DECIMAL(10,6),

    -- Trigger
    triggered_by VARCHAR(50),  -- evidence, feedback, admin_update, scheduled
    trigger_details JSONB,

    -- Audit
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- ============================================================================
-- 39.2.5 Indexes
-- ============================================================================

-- Need patterns indexes
CREATE INDEX idx_neural_need_patterns_tenant ON neural_need_patterns(tenant_id);
CREATE INDEX idx_neural_need_patterns_status ON neural_need_patterns(status);
CREATE INDEX idx_neural_need_patterns_hash ON neural_need_patterns(pattern_hash);
CREATE INDEX idx_neural_need_patterns_score ON neural_need_patterns(total_evidence_score DES
CREATE INDEX idx_neural_need_patterns_embedding ON neural_need_patterns USING ivfflat (patte

-- Evidence indexes
CREATE INDEX idx_neural_need_evidence_pattern ON neural_need_evidence(pattern_id);
CREATE INDEX idx_neural_need_evidence_user ON neural_need_evidence(user_id);
CREATE INDEX idx_neural_need_evidence_type ON neural_need_evidence(evidence_type);
CREATE INDEX idx_neural_need_evidence_occurred ON neural_need_evidence(occurred_at DESC);

-- Proposal indexes
CREATE INDEX idx_workflow_proposals_tenant ON workflow_proposals(tenant_id);
CREATE INDEX idx_workflow_proposals_status ON workflow_proposals(admin_status);
CREATE INDEX idx_workflow_proposals_pattern ON workflow_proposals(source_pattern_id);
CREATE INDEX idx_workflow_proposals_created ON workflow_proposals(created_at DESC);

-- Review indexes
CREATE INDEX idx_workflow_proposal_reviews_proposal ON workflow_proposal_reviews(proposal_id
CREATE INDEX idx_workflow_proposal_reviews_reviewer ON workflow_proposal_reviews(reviewer_ty

-- Config indexes
CREATE INDEX idx_proposal_threshold_config_tenant ON proposal_threshold_config(tenant_id);
CREATE INDEX idx_evidence_weight_config_tenant ON evidence_weight_config(tenant_id);
```

```sql
-- Learning session indexes
CREATE INDEX idx_neural_learning_sessions_tenant ON neural_learning_sessions(tenant_id);
CREATE INDEX idx_neural_learning_sessions_type ON neural_learning_sessions(session_type);
CREATE INDEX idx_neural_embedding_updates_entity ON neural_embedding_updates(entity_type, en

-- ============================================================================
-- 39.2.6 Row Level Security
-- ============================================================================

ALTER TABLE neural_need_patterns ENABLE ROW LEVEL SECURITY;
ALTER TABLE neural_need_evidence ENABLE ROW LEVEL SECURITY;
ALTER TABLE workflow_proposals ENABLE ROW LEVEL SECURITY;
ALTER TABLE workflow_proposal_reviews ENABLE ROW LEVEL SECURITY;
ALTER TABLE proposal_threshold_config ENABLE ROW LEVEL SECURITY;
ALTER TABLE evidence_weight_config ENABLE ROW LEVEL SECURITY;
ALTER TABLE neural_learning_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE neural_embedding_updates ENABLE ROW LEVEL SECURITY;

-- Policies (tenant isolation)
CREATE POLICY tenant_isolation_neural_need_patterns ON neural_need_patterns
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tenant_isolation_neural_need_evidence ON neural_need_evidence
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tenant_isolation_workflow_proposals ON workflow_proposals
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tenant_isolation_workflow_proposal_reviews ON workflow_proposal_reviews
    USING (proposal_id IN (SELECT id FROM workflow_proposals WHERE tenant_id = current_setti

CREATE POLICY tenant_isolation_proposal_threshold_config ON proposal_threshold_config
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tenant_isolation_evidence_weight_config ON evidence_weight_config
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tenant_isolation_neural_learning_sessions ON neural_learning_sessions
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tenant_isolation_neural_embedding_updates ON neural_embedding_updates
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

-- ============================================================================
-- 39.2.7 Default Data Seeding
-- ============================================================================
```

```sql
-- Seed default evidence weights (will be copied per tenant on creation)
INSERT INTO evidence_weight_config (tenant_id, evidence_type, weight, description, enabled)
SELECT
    t.id,
    e.evidence_type,
    e.weight,
    e.description,
    TRUE
FROM tenants t
CROSS JOIN (VALUES
    ('workflow_failure', 0.40, 'Existing workflow failed to complete'),
    ('negative_feedback', 0.35, 'User gave thumbs down or negative rating'),
    ('manual_override', 0.15, 'User manually switched model/approach'),
    ('regenerate_request', 0.10, 'User requested regeneration'),
    ('abandon_session', 0.20, 'User abandoned mid-conversation'),
    ('low_confidence_completion', 0.15, 'Workflow completed but Neural confidence < 0.5'),
    ('explicit_request', 0.50, 'User explicitly requested different approach')
) AS e(evidence_type, weight, description)
ON CONFLICT (tenant_id, evidence_type) DO NOTHING;

-- Seed default threshold config per tenant
INSERT INTO proposal_threshold_config (tenant_id)
SELECT id FROM tenants
ON CONFLICT (tenant_id) DO NOTHING;
```

39.3 TypeScript Types

```typescript
// ============================================================================
// packages/core/src/types/workflow-proposals.ts
// Dynamic Workflow Proposal System Types
// ============================================================================

import { UUID, Timestamp, TenantId, UserId } from './common';

// ============================================================================
// 39.3.1 Evidence Types
// ============================================================================

export type EvidenceType =
  | 'workflow_failure'
  | 'negative_feedback'
  | 'manual_override'
  | 'regenerate_request'
```

```
  | 'abandon_session'
  | 'low_confidence_completion'
  | 'explicit_request';

export interface EvidenceWeightConfig {
  id: UUID;
  tenantId: TenantId;
  evidenceType: EvidenceType;
  weight: number;
  description: string;
  enabled: boolean;
  createdAt: Timestamp;
  updatedAt: Timestamp;
  updatedBy?: UserId;
}

export interface NeedEvidence {
  id: UUID;
  tenantId: TenantId;
  patternId: UUID;
  userId?: UserId;
  sessionId?: UUID;
  executionId?: UUID;
  evidenceType: EvidenceType;
  evidenceWeight: number;
  evidenceData: Record<string, unknown>;
  originalRequest?: string;
  attemptedWorkflowId?: UUID;
  failureReason?: string;
  userFeedback?: string;
  occurredAt: Timestamp;
  createdAt: Timestamp;
}

// ============================================================================
// 39.3.2 Need Pattern Types
// ============================================================================

export type PatternStatus =
  | 'accumulating'
  | 'threshold_met'
  | 'proposal_generated'
  | 'resolved';

export interface PatternSignature {
  intentCategory: string;
```

```
  keywords: string[];
  domainHints: string[];
  failedWorkflowTypes: string[];
  userSegments: string[];
  contextualFactors: Record<string, unknown>;
}

export interface NeedPattern {
  id: UUID;
  tenantId: TenantId;
  patternHash: string;
  patternSignature: PatternSignature;
  patternEmbedding?: number[];  // 768-dim vector
  patternName: string;
  patternDescription?: string;
  detectedIntent: string;
  existingWorkflowGaps: string[];
  totalEvidenceScore: number;
  evidenceCount: number;
  uniqueUsersAffected: number;
  firstOccurrence: Timestamp;
  lastOccurrence: Timestamp;
  occurrenceThresholdMet: boolean;
  impactThresholdMet: boolean;
  confidenceThresholdMet: boolean;
  status: PatternStatus;
  proposalId?: UUID;
  createdAt: Timestamp;
  updatedAt: Timestamp;
}

export interface PatternWithEvidence extends NeedPattern {
  evidence: NeedEvidence[];
  evidenceSummary: {
    byType: Record<EvidenceType, number>;
    byUser: Record<string, number>;
    timeline: Array<{ date: string; count: number; score: number }>;
  };
}

// ============================================================================
// 39.3.3 Proposal Types
// ============================================================================

export type ProposalAdminStatus =
  | 'pending_brain'
```

```
  | 'pending_admin'
  | 'approved'
  | 'declined'
  | 'testing';

export type ProposalTestStatus =
  | 'not_tested'
  | 'testing'
  | 'passed'
  | 'failed';

export type WorkflowComplexity =
  | 'simple'
  | 'moderate'
  | 'complex'
  | 'advanced';

export type ProposedWorkflowType =
  | 'orchestration_pattern'
  | 'production_workflow'
  | 'hybrid';

export interface BrainRiskAssessment {
  costRisk: number;
  latencyRisk: number;
  qualityRisk: number;
  complianceRisk: number;
  overallRisk: number;
  riskFactors: string[];
  mitigations: string[];
}

export interface ProposedWorkflowNode {
  id: string;
  type: string;
  name: string;
  config: Record<string, unknown>;
  position: { x: number; y: number };
  connections: string[];
}

export interface ProposedWorkflowDAG {
  nodes: ProposedWorkflowNode[];
  edges: Array<{ from: string; to: string; condition?: string }>;
  entryPoint: string;
  exitPoints: string[];
```

```
  metadata: {
    estimatedLatencyMs: number;
    estimatedCostPer1k: number;
    requiredModels: string[];
    requiredServices: string[];
  };
}

export interface WorkflowProposal {
  id: UUID;
  tenantId: TenantId;
  proposalCode: string;
  proposalName: string;
  proposalDescription: string;
  sourcePatternId: UUID;
  proposedWorkflow: ProposedWorkflowDAG;
  workflowCategory?: string;
  workflowType: ProposedWorkflowType;
  estimatedComplexity: WorkflowComplexity;

  // Neural Engine analysis
  neuralConfidence: number;
  neuralReasoning: string;
  estimatedQualityImprovement?: number;
  estimatedCoveragePercentage?: number;

  // Brain Governor review
  brainApproved?: boolean;
  brainReviewTimestamp?: Timestamp;
  brainRiskAssessment?: BrainRiskAssessment;
  brainVetoReason?: string;
  brainModifications?: Partial<ProposedWorkflowDAG>;

  // Evidence summary
  evidenceCount: number;
  uniqueUsersAffected: number;
  totalEvidenceScore: number;
  evidenceTimeSpanDays: number;
  evidenceSummary: {
    byType: Record<EvidenceType, number>;
    topFailureReasons: string[];
    affectedDomains: string[];
  };

  // Admin review
  adminStatus: ProposalAdminStatus;
```

```
  adminReviewerId?: UserId;
  adminReviewTimestamp?: Timestamp;
  adminNotes?: string;
  adminModifications?: Partial<ProposedWorkflowDAG>;

  // Testing
  testStatus?: ProposalTestStatus;
  testResults?: {
    testsRun: number;
    testsPassed: number;
    testsFailed: number;
    avgLatencyMs: number;
    avgQualityScore: number;
    issues: string[];
  };
  testExecutionIds?: UUID[];

  // Publishing
  publishedWorkflowId?: UUID;
  publishedAt?: Timestamp;

  // Audit
  proposalBatchId?: UUID;
  createdAt: Timestamp;
  updatedAt: Timestamp;
}

export interface ProposalWithPattern extends WorkflowProposal {
  sourcePattern: NeedPattern;
}

// ============================================================================
// 39.3.4 Review Types
// ============================================================================

export type ReviewerType = 'brain' | 'admin';

export type ReviewAction =
  | 'approve'
  | 'decline'
  | 'modify'
  | 'request_test'
  | 'escalate';

export interface ProposalReview {
  id: UUID;
```

```
  proposalId: UUID;
  reviewerType: ReviewerType;
  reviewerId?: UserId;
  action: ReviewAction;
  previousStatus: ProposalAdminStatus;
  newStatus: ProposalAdminStatus;
  reviewNotes?: string;
  modificationsMade?: Partial<ProposedWorkflowDAG>;
  riskAssessment?: BrainRiskAssessment;
  reviewedAt: Timestamp;
}

// ============================================================================
// 39.3.5 Threshold Configuration Types
// ============================================================================

export interface ProposalThresholdConfig {
  id: UUID;
  tenantId: TenantId;

  // Occurrence thresholds
  minEvidenceCount: number;
  minUniqueUsers: number;
  minTimeSpanHours: number;
  maxTimeSpanDays: number;

  // Impact thresholds
  minTotalEvidenceScore: number;
  minAvgEvidenceWeight: number;

  // Neural confidence thresholds
  minNeuralConfidence: number;
  minCoverageEstimate: number;

  // Brain approval thresholds
  maxCostRisk: number;
  maxLatencyRisk: number;
  minQualityConfidence: number;
  maxComplianceRisk: number;

  // Rate limiting
  maxProposalsPerDay: number;
  maxProposalsPerWeek: number;
  cooldownAfterDeclineHours: number;

  // Auto-approve settings
```

```typescript
  autoApproveEnabled: boolean;
  autoApproveMinConfidence: number;
  autoApproveMaxComplexity: WorkflowComplexity;

  // Audit
  createdAt: Timestamp;
  updatedAt: Timestamp;
  updatedBy?: UserId;
}

// ============================================================================
// 39.3.6 API Request/Response Types
// ============================================================================

// Evidence submission
export interface SubmitEvidenceRequest {
  evidenceType: EvidenceType;
  sessionId?: UUID;
  executionId?: UUID;
  originalRequest?: string;
  attemptedWorkflowId?: UUID;
  failureReason?: string;
  userFeedback?: string;
  evidenceData?: Record<string, unknown>;
}

export interface SubmitEvidenceResponse {
  evidenceId: UUID;
  patternId: UUID;
  patternStatus: PatternStatus;
  currentEvidenceScore: number;
  thresholdsMet: {
    occurrence: boolean;
    impact: boolean;
    confidence: boolean;
  };
}

// Pattern queries
export interface GetPatternsRequest {
  status?: PatternStatus[];
  minEvidenceScore?: number;
  minEvidenceCount?: number;
  limit?: number;
  offset?: number;
}
```

```typescript
export interface GetPatternsResponse {
  patterns: PatternWithEvidence[];
  total: number;
  hasMore: boolean;
}

// Proposal queries
export interface GetProposalsRequest {
  status?: ProposalAdminStatus[];
  testStatus?: ProposalTestStatus[];
  minConfidence?: number;
  limit?: number;
  offset?: number;
}

export interface GetProposalsResponse {
  proposals: ProposalWithPattern[];
  total: number;
  hasMore: boolean;
}

// Admin review
export interface ReviewProposalRequest {
  action: ReviewAction;
  notes?: string;
  modifications?: Partial<ProposedWorkflowDAG>;
}

export interface ReviewProposalResponse {
  proposalId: UUID;
  newStatus: ProposalAdminStatus;
  reviewId: UUID;
}

// Testing
export interface TestProposalRequest {
  testMode: 'manual' | 'automated';
  testCases?: Array<{
    input: string;
    expectedBehavior?: string;
  }>;
  iterations?: number;
}

export interface TestProposalResponse {
```

```
  proposalId: UUID;
  testStatus: ProposalTestStatus;
  testResults: WorkflowProposal['testResults'];
  testExecutionIds: UUID[];
}

// Publishing
export interface PublishProposalRequest {
  publishToCategory?: string;
  overrideWorkflowId?: UUID;  // Replace existing workflow
}

export interface PublishProposalResponse {
  proposalId: UUID;
  publishedWorkflowId: UUID;
  publishedAt: Timestamp;
}

// Configuration updates
export interface UpdateThresholdConfigRequest {
  config: Partial<Omit<ProposalThresholdConfig, 'id' | 'tenantId' | 'createdAt' | 'updatedAt
}

export interface UpdateEvidenceWeightsRequest {
  weights: Array<{
    evidenceType: EvidenceType;
    weight: number;
    enabled?: boolean;
  }>;
}

// ============================================================================
// 39.3.7 Neural Engine Types (for proposal generation)
// ============================================================================

export interface NeuralProposalGenerationContext {
  pattern: NeedPattern;
  evidence: NeedEvidence[];
  existingWorkflows: Array<{
    id: UUID;
    name: string;
    similarity: number;
    failureRate: number;
  }>;
  userSegmentProfile: {
    commonIntents: string[];
```

```
    preferredComplexity: WorkflowComplexity;
    domainDistribution: Record<string, number>;
  };
  tenantConstraints: {
    maxWorkflowNodes: number;
    allowedNodeTypes: string[];
    requiredComplianceChecks: string[];
  };
}

export interface NeuralProposalGenerationResult {
  success: boolean;
  proposal?: Omit<WorkflowProposal, 'id' | 'tenantId' | 'createdAt' | 'updatedAt'>;
  confidence: number;
  reasoning: string;
  alternativeApproaches?: Array<{
    description: string;
    confidence: number;
    tradeoffs: string[];
  }>;
  rejectionReason?: string;
}

// ============================================================================
// 39.3.8 Brain Governor Types (for proposal review)
// ============================================================================

export interface BrainProposalReviewContext {
  proposal: WorkflowProposal;
  pattern: NeedPattern;
  tenantConfig: ProposalThresholdConfig;
  currentWorkflowCount: number;
  recentProposalCount: {
    today: number;
    thisWeek: number;
  };
  similarExistingWorkflows: Array<{
    id: UUID;
    name: string;
    similarity: number;
  }>;
}

export interface BrainProposalReviewResult {
  approved: boolean;
  riskAssessment: BrainRiskAssessment;
```

```
  reasoning: string;
  vetoReason?: string;
  suggestedModifications?: Partial<ProposedWorkflowDAG>;
  escalateToAdmin: boolean;
  adminPriority?: 'low' | 'medium' | 'high' | 'urgent';
}
```

39.4 Constants and Defaults

```
// ============================================================================
// packages/core/src/constants/workflow-proposals.ts
// ============================================================================

import { EvidenceType, WorkflowComplexity } from '../types/workflow-proposals';

// Default evidence weights
export const DEFAULT_EVIDENCE_WEIGHTS: Record<EvidenceType, number> = {
  workflow_failure: 0.40,
  negative_feedback: 0.35,
  manual_override: 0.15,
  regenerate_request: 0.10,
  abandon_session: 0.20,
  low_confidence_completion: 0.15,
  explicit_request: 0.50,
};

// Default threshold configuration
export const DEFAULT_THRESHOLD_CONFIG = {
  // Occurrence thresholds
  minEvidenceCount: 5,
  minUniqueUsers: 3,
  minTimeSpanHours: 24,
  maxTimeSpanDays: 30,

  // Impact thresholds
  minTotalEvidenceScore: 0.60,
  minAvgEvidenceWeight: 0.20,

  // Neural confidence thresholds
  minNeuralConfidence: 0.75,
  minCoverageEstimate: 0.60,

  // Brain approval thresholds
  maxCostRisk: 0.30,
```

```
  maxLatencyRisk: 0.40,
  minQualityConfidence: 0.70,
  maxComplianceRisk: 0.10,

  // Rate limiting
  maxProposalsPerDay: 10,
  maxProposalsPerWeek: 30,
  cooldownAfterDeclineHours: 72,

  // Auto-approve (disabled by default)
  autoApproveEnabled: false,
  autoApproveMinConfidence: 0.95,
  autoApproveMaxComplexity: 'simple' as WorkflowComplexity,
};

// Complexity scoring
export const COMPLEXITY_NODE_THRESHOLDS: Record<WorkflowComplexity, { min: number; max: numb
  simple: { min: 1, max: 3 },
  moderate: { min: 4, max: 7 },
  complex: { min: 8, max: 12 },
  advanced: { min: 13, max: Infinity },
};

// Proposal code generation
export const PROPOSAL_CODE_PREFIX = 'WP';
export const PROPOSAL_CODE_YEAR_FORMAT = 'YYYY';
export const PROPOSAL_CODE_SEQUENCE_PADDING = 3;

// Rate limit windows
export const RATE_LIMIT_WINDOWS = {
  daily: 24 * 60 * 60 * 1000,   // 24 hours in ms
  weekly: 7 * 24 * 60 * 60 * 1000,  // 7 days in ms
};

// Pattern similarity threshold for deduplication
export const PATTERN_SIMILARITY_THRESHOLD = 0.85;

// Maximum nodes in a proposed workflow
export const MAX_PROPOSED_WORKFLOW_NODES = 20;

// Test configuration defaults
export const DEFAULT_TEST_CONFIG = {
  automatedIterations: 10,
  timeoutMs: 30000,
  minPassRate: 0.80,
};
```

This chunk contains: - Complete database schema (8 tables with RLS) - Full TypeScript type definitions - Constants and default configurations

Next chunk will contain Lambda functions for evidence collection, pattern detection, and proposal generation.

39.5 Evidence Collection Lambda

```
// ============================================================================
// packages/lambdas/src/workflow-proposals/evidence-collector.ts
// Collects and processes evidence of user needs not met by existing workflows
// ============================================================================

import { APIGatewayProxyHandler, APIGatewayProxyResult } from 'aws-lambda';
import { Pool } from 'pg';
import { createHash } from 'crypto';
import {
  EvidenceType,
  NeedEvidence,
  NeedPattern,
  PatternSignature,
  SubmitEvidenceRequest,
  SubmitEvidenceResponse,
} from '@radiant/core/types/workflow-proposals';
import { DEFAULT_EVIDENCE_WEIGHTS, PATTERN_SIMILARITY_THRESHOLD } from '@radiant/core/consta
import { getTenantId, getUserId, createResponse, logAudit } from '../utils';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  max: 10,
});

// ============================================================================
// Pattern Signature Generation
// ============================================================================

interface PatternContext {
  originalRequest: string;
  attemptedWorkflowId?: string;
  failureReason?: string;
  evidenceType: EvidenceType;
  evidenceData: Record<string, unknown>;
}
```

```
function generatePatternSignature(context: PatternContext): PatternSignature {
  // Extract keywords from request
  const keywords = extractKeywords(context.originalRequest || '');

  // Determine intent category from failure and request
  const intentCategory = classifyIntent(context);

  // Extract domain hints
  const domainHints = extractDomainHints(context);

  // Track failed workflow types
  const failedWorkflowTypes = context.attemptedWorkflowId
    ? [context.attemptedWorkflowId]
    : [];

  return {
    intentCategory,
    keywords,
    domainHints,
    failedWorkflowTypes,
    userSegments: [],  // Populated during pattern analysis
    contextualFactors: {
      evidenceType: context.evidenceType,
      hasExplicitFeedback: !!context.failureReason,
    },
  };
}

function extractKeywords(text: string): string[] {
  // Simple keyword extraction - in production, use NLP service
  const stopWords = new Set(['the', 'a', 'an', 'is', 'are', 'was', 'were', 'be', 'been',
    'being', 'have', 'has', 'had', 'do', 'does', 'did', 'will', 'would', 'could', 'should',
    'may', 'might', 'must', 'shall', 'can', 'need', 'dare', 'ought', 'used', 'to', 'of',
    'in', 'for', 'on', 'with', 'at', 'by', 'from', 'as', 'into', 'through', 'during',
    'before', 'after', 'above', 'below', 'between', 'under', 'again', 'further', 'then',
    'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'each', 'few', 'more',
    'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
    'than', 'too', 'very', 'just', 'and', 'but', 'if', 'or', 'because', 'until', 'while',
    'although', 'though', 'after', 'before', 'when', 'i', 'me', 'my', 'myself', 'we',
    'our', 'you', 'your', 'he', 'him', 'she', 'her', 'it', 'its', 'they', 'them', 'what',
    'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'was',
    'were', 'been', 'being', 'please', 'help', 'want', 'like', 'get', 'make']);

  return text
    .toLowerCase()
```

```
    .replace(/[^\w\s]/g, ' ')
    .split(/\s+/)
    .filter(word => word.length > 2 && !stopWords.has(word))
    .slice(0, 20);  // Limit to top 20 keywords
}

function classifyIntent(context: PatternContext): string {
  const request = (context.originalRequest || '').toLowerCase();
  const failure = (context.failureReason || '').toLowerCase();

  // Intent classification based on patterns
  const intentPatterns: Record<string, string[]> = {
    'research': ['research', 'find', 'search', 'look up', 'investigate', 'explore'],
    'analysis': ['analyze', 'analysis', 'examine', 'evaluate', 'assess', 'review'],
    'creation': ['create', 'write', 'generate', 'make', 'build', 'compose', 'draft'],
    'comparison': ['compare', 'versus', 'vs', 'difference', 'contrast', 'better'],
    'explanation': ['explain', 'how', 'why', 'what is', 'describe', 'clarify'],
    'transformation': ['convert', 'transform', 'translate', 'change', 'modify'],
    'summarization': ['summarize', 'summary', 'brief', 'tldr', 'key points'],
    'verification': ['verify', 'check', 'validate', 'confirm', 'fact-check'],
    'coding': ['code', 'program', 'function', 'script', 'debug', 'fix bug'],
    'data_processing': ['data', 'csv', 'json', 'parse', 'extract', 'process'],
  };

  for (const [intent, patterns] of Object.entries(intentPatterns)) {
    if (patterns.some(p => request.includes(p) || failure.includes(p))) {
      return intent;
    }
  }

  return 'general';
}

function extractDomainHints(context: PatternContext): string[] {
  const text = `${context.originalRequest || ''} ${context.failureReason || ''}`.toLowerCase

  const domainPatterns: Record<string, string[]> = {
    'medical': ['medical', 'health', 'diagnosis', 'symptom', 'treatment', 'patient', 'doctor
    'legal': ['legal', 'law', 'contract', 'court', 'attorney', 'regulation', 'compliance'],
    'financial': ['financial', 'money', 'investment', 'stock', 'budget', 'accounting', 'tax'
    'scientific': ['scientific', 'research', 'experiment', 'hypothesis', 'study', 'paper'],
    'technical': ['technical', 'engineering', 'software', 'hardware', 'system', 'architectur
    'creative': ['creative', 'story', 'poem', 'art', 'design', 'music', 'novel'],
    'educational': ['educational', 'learn', 'teach', 'course', 'student', 'lesson', 'tutoria
    'business': ['business', 'company', 'strategy', 'market', 'sales', 'customer', 'product'
  };
```

```
  const hints: string[] = [];
  for (const [domain, patterns] of Object.entries(domainPatterns)) {
    if (patterns.some(p => text.includes(p))) {
      hints.push(domain);
    }
  }

  return hints;
}

function generatePatternHash(signature: PatternSignature): string {
  // Normalize and hash the signature for deduplication
  const normalized = {
    intent: signature.intentCategory,
    keywords: [...signature.keywords].sort(),
    domains: [...signature.domainHints].sort(),
  };

  return createHash('sha256')
    .update(JSON.stringify(normalized))
    .digest('hex');
}

// ============================================================================
// Pattern Embedding Generation (calls Neural Engine)
// ============================================================================

async function generatePatternEmbedding(signature: PatternSignature): Promise<number[]> {
  // In production, call the Neural Engine embedding service
  // For now, return a placeholder that would be replaced by actual embedding
  const text = [
    signature.intentCategory,
    ...signature.keywords,
    ...signature.domainHints,
  ].join(' ');

  // This would call: POST /api/v2/internal/neural/embed
  // return await neuralEngineClient.generateEmbedding(text);

  // Placeholder: return 768-dim zero vector (will be populated by Neural Engine)
  return new Array(768).fill(0);
}

// ============================================================================
// Evidence Weight Resolution
```

```typescript
// ============================================================================

async function getEvidenceWeight(
  client: Pool,
  tenantId: string,
  evidenceType: EvidenceType
): Promise<number> {
  const result = await client.query(
    `SELECT weight FROM evidence_weight_config
     WHERE tenant_id = $1 AND evidence_type = $2 AND enabled = TRUE`,
    [tenantId, evidenceType]
  );

  if (result.rows.length > 0) {
    return parseFloat(result.rows[0].weight);
  }

  return DEFAULT_EVIDENCE_WEIGHTS[evidenceType] || 0.10;
}

// ============================================================================
// Pattern Finding/Creation
// ============================================================================

async function findOrCreatePattern(
  client: Pool,
  tenantId: string,
  signature: PatternSignature,
  detectedIntent: string
): Promise<{ pattern: NeedPattern; isNew: boolean }> {
  const patternHash = generatePatternHash(signature);

  // Try to find existing pattern
  const existing = await client.query<NeedPattern>(
    `SELECT * FROM neural_need_patterns
     WHERE tenant_id = $1 AND pattern_hash = $2`,
    [tenantId, patternHash]
  );

  if (existing.rows.length > 0) {
    return { pattern: existing.rows[0], isNew: false };
  }

  // Check for similar patterns using embedding similarity
  const embedding = await generatePatternEmbedding(signature);
```

```
  const similar = await client.query<NeedPattern>(
    `SELECT *, 1 - (pattern_embedding <=> $2::vector) as similarity
     FROM neural_need_patterns
     WHERE tenant_id = $1
       AND pattern_embedding IS NOT NULL
       AND 1 - (pattern_embedding <=> $2::vector) > $3
     ORDER BY similarity DESC
     LIMIT 1`,
    [tenantId, JSON.stringify(embedding), PATTERN_SIMILARITY_THRESHOLD]
  );

  if (similar.rows.length > 0) {
    // Merge into existing similar pattern
    return { pattern: similar.rows[0], isNew: false };
  }

  // Create new pattern
  const patternName = `${signature.intentCategory}_${signature.keywords.slice(0, 3).join('_'

  const inserted = await client.query<NeedPattern>(
    `INSERT INTO neural_need_patterns (
      tenant_id, pattern_hash, pattern_signature, pattern_embedding,
      pattern_name, detected_intent, existing_workflow_gaps
    ) VALUES ($1, $2, $3, $4::vector, $5, $6, $7)
    RETURNING *`,
    [
      tenantId,
      patternHash,
      JSON.stringify(signature),
      JSON.stringify(embedding),
      patternName,
      detectedIntent,
      signature.failedWorkflowTypes,
    ]
  );

  return { pattern: inserted.rows[0], isNew: true };
}

// ============================================================================
// Threshold Checking
// ============================================================================

interface ThresholdStatus {
  occurrence: boolean;
  impact: boolean;
```

```
  confidence: boolean;
  allMet: boolean;
}

async function checkThresholds(
  client: Pool,
  tenantId: string,
  pattern: NeedPattern
): Promise<ThresholdStatus> {
  // Get tenant threshold config
  const configResult = await client.query(
    `SELECT * FROM proposal_threshold_config WHERE tenant_id = $1`,
    [tenantId]
  );

  const config = configResult.rows[0] || {};
  const minEvidenceCount = config.min_evidence_count || 5;
  const minUniqueUsers = config.min_unique_users || 3;
  const minTimeSpanHours = config.min_time_span_hours || 24;
  const minTotalScore = parseFloat(config.min_total_evidence_score || '0.60');

  // Calculate time span
  const timeSpanHours = pattern.last_occurrence && pattern.first_occurrence
    ? (new Date(pattern.last_occurrence).getTime() - new Date(pattern.first_occurrence).getT
    : 0;

  const occurrence =
    pattern.evidence_count >= minEvidenceCount &&
    pattern.unique_users_affected >= minUniqueUsers &&
    timeSpanHours >= minTimeSpanHours;

  const impact = pattern.total_evidence_score >= minTotalScore;

  // Confidence threshold requires Neural Engine analysis (checked during proposal generati
  const confidence = pattern.confidence_threshold_met || false;

  return {
    occurrence,
    impact,
    confidence,
    allMet: occurrence && impact && confidence,
  };
}

// ============================================================================
// Main Handler
```

```
// ============================================================================

export const handler: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxyResult>
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);

    if (!tenantId || !userId) {
      return createResponse(401, { error: 'Unauthorized' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    const request: SubmitEvidenceRequest = JSON.parse(event.body || '{}');

    if (!request.evidenceType) {
      return createResponse(400, { error: 'evidenceType is required' });
    }

    // Get evidence weight
    const evidenceWeight = await getEvidenceWeight(client, tenantId, request.evidenceType);

    // Generate pattern signature
    const signature = generatePatternSignature({
      originalRequest: request.originalRequest || '',
      attemptedWorkflowId: request.attemptedWorkflowId,
      failureReason: request.failureReason,
      evidenceType: request.evidenceType,
      evidenceData: request.evidenceData || {},
    });

    // Find or create pattern
    const { pattern, isNew } = await findOrCreatePattern(
      client,
      tenantId,
      signature,
      classifyIntent({
        originalRequest: request.originalRequest || '',
        failureReason: request.failureReason,
        evidenceType: request.evidenceType,
        evidenceData: request.evidenceData || {},
      })
    );
```

```
// Insert evidence record
const evidenceResult = await client.query<NeedEvidence>(
  `INSERT INTO neural_need_evidence (
    tenant_id, pattern_id, user_id, session_id, execution_id,
    evidence_type, evidence_weight, evidence_data,
    original_request, attempted_workflow_id, failure_reason, user_feedback
  ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12)
  RETURNING id`,
  [
    tenantId,
    pattern.id,
    userId,
    request.sessionId,
    request.executionId,
    request.evidenceType,
    evidenceWeight,
    JSON.stringify(request.evidenceData || {}),
    request.originalRequest,
    request.attemptedWorkflowId,
    request.failureReason,
    request.userFeedback,
  ]
);

// Update pattern statistics
await client.query(
  `UPDATE neural_need_patterns SET
    total_evidence_score = total_evidence_score + $2,
    evidence_count = evidence_count + 1,
    unique_users_affected = (
      SELECT COUNT(DISTINCT user_id)
      FROM neural_need_evidence
      WHERE pattern_id = $1
    ),
    last_occurrence = NOW(),
    updated_at = NOW()
  WHERE id = $1`,
  [pattern.id, evidenceWeight]
);

// Fetch updated pattern
const updatedPattern = await client.query<NeedPattern>(
  `SELECT * FROM neural_need_patterns WHERE id = $1`,
  [pattern.id]
);
```

```
    // Check thresholds
    const thresholds = await checkThresholds(client, tenantId, updatedPattern.rows[0]);

    // Update threshold status
    await client.query(
      `UPDATE neural_need_patterns SET
        occurrence_threshold_met = $2,
        impact_threshold_met = $3,
        status = CASE
          WHEN $2 AND $3 THEN 'threshold_met'
          ELSE status
        END
      WHERE id = $1`,
      [pattern.id, thresholds.occurrence, thresholds.impact]
    );

    // Audit log
    await logAudit(client, {
      tenantId,
      userId,
      action: 'evidence_submitted',
      resourceType: 'need_pattern',
      resourceId: pattern.id,
      details: {
        evidenceId: evidenceResult.rows[0].id,
        evidenceType: request.evidenceType,
        evidenceWeight,
        thresholdsMet: thresholds,
        isNewPattern: isNew,
      },
    });

    const response: SubmitEvidenceResponse = {
      evidenceId: evidenceResult.rows[0].id,
      patternId: pattern.id,
      patternStatus: thresholds.occurrence && thresholds.impact ? 'threshold_met' : 'accumul
      currentEvidenceScore: updatedPattern.rows[0].total_evidence_score + evidenceWeight,
      thresholdsMet: thresholds,
    };

    return createResponse(200, response);

  } catch (error) {
    console.error('Evidence collection error:', error);
    return createResponse(500, { error: 'Internal server error' });
  } finally {
```

```
    client.release();
  }
};
```

39.6 Pattern Analysis Lambda (Scheduled)

```
// ============================================================================
// packages/lambdas/src/workflow-proposals/pattern-analyzer.ts
// Scheduled Lambda that analyzes patterns and triggers proposal generation
// ============================================================================

import { ScheduledHandler } from 'aws-lambda';
import { Pool } from 'pg';
import { SQSClient, SendMessageCommand } from '@aws-sdk/client-sqs';
import { NeedPattern, PatternWithEvidence } from '@radiant/core/types/workflow-proposals';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  max: 10,
});

const sqs = new SQSClient({});
const PROPOSAL_QUEUE_URL = process.env.PROPOSAL_QUEUE_URL!;

// ============================================================================
// Pattern Analysis
// ============================================================================

interface AnalysisResult {
  patternsAnalyzed: number;
  patternsQualified: number;
  proposalsQueued: number;
  errors: string[];
}

async function analyzePatterns(tenantId: string): Promise<AnalysisResult> {
  const client = await pool.connect();
  const result: AnalysisResult = {
    patternsAnalyzed: 0,
    patternsQualified: 0,
    proposalsQueued: 0,
    errors: [],
  };
```

```
try {
  await client.query(`SET app.current_tenant_id = '${tenantId}'`);

  // Get threshold config
  const configResult = await client.query(
    `SELECT * FROM proposal_threshold_config WHERE tenant_id = $1`,
    [tenantId]
  );
  const config = configResult.rows[0] || {};

  // Check rate limits
  const rateLimitResult = await client.query(
    `SELECT
      COUNT(*) FILTER (WHERE created_at > NOW() - INTERVAL '1 day') as today,
      COUNT(*) FILTER (WHERE created_at > NOW() - INTERVAL '7 days') as this_week
     FROM workflow_proposals
     WHERE tenant_id = $1`,
    [tenantId]
  );

  const dailyCount = parseInt(rateLimitResult.rows[0].today);
  const weeklyCount = parseInt(rateLimitResult.rows[0].this_week);
  const maxDaily = config.max_proposals_per_day || 10;
  const maxWeekly = config.max_proposals_per_week || 30;

  if (dailyCount >= maxDaily || weeklyCount >= maxWeekly) {
    result.errors.push(`Rate limit reached: ${dailyCount}/${maxDaily} daily, ${weeklyCount
    return result;
  }

  const remainingDaily = maxDaily - dailyCount;
  const remainingWeekly = maxWeekly - weeklyCount;
  const maxProposals = Math.min(remainingDaily, remainingWeekly);

  // Find patterns that have met thresholds but don't have proposals yet
  const patterns = await client.query<NeedPattern>(
    `SELECT * FROM neural_need_patterns
     WHERE tenant_id = $1
       AND status = 'threshold_met'
       AND occurrence_threshold_met = TRUE
       AND impact_threshold_met = TRUE
       AND proposal_id IS NULL
     ORDER BY total_evidence_score DESC
     LIMIT $2`,
    [tenantId, maxProposals]
  );
```

```
result.patternsAnalyzed = patterns.rows.length;

for (const pattern of patterns.rows) {
  try {
    // Get evidence for pattern
    const evidenceResult = await client.query(
      `SELECT * FROM neural_need_evidence
       WHERE pattern_id = $1
       ORDER BY occurred_at DESC
       LIMIT 100`,
      [pattern.id]
    );

    // Check for recent declines (cooldown period)
    const cooldownHours = config.cooldown_after_decline_hours || 72;
    const recentDecline = await client.query(
      `SELECT 1 FROM workflow_proposals wp
       JOIN workflow_proposal_reviews wpr ON wpr.proposal_id = wp.id
       WHERE wp.source_pattern_id = $1
         AND wpr.action = 'decline'
         AND wpr.reviewed_at > NOW() - INTERVAL '${cooldownHours} hours'
       LIMIT 1`,
      [pattern.id]
    );

    if (recentDecline.rows.length > 0) {
      result.errors.push(`Pattern ${pattern.id} in cooldown after recent decline`);
      continue;
    }

    // Queue for proposal generation
    const message: PatternWithEvidence = {
      ...pattern,
      evidence: evidenceResult.rows,
      evidenceSummary: generateEvidenceSummary(evidenceResult.rows),
    };

    await sqs.send(new SendMessageCommand({
      QueueUrl: PROPOSAL_QUEUE_URL,
      MessageBody: JSON.stringify({
        type: 'generate_proposal',
        tenantId,
        pattern: message,
      }),
      MessageGroupId: tenantId,
```

```
            MessageDeduplicationId: `${pattern.id}-${Date.now()}`,
          }));

          result.patternsQualified++;
          result.proposalsQueued++;

          // Update pattern status
          await client.query(
            `UPDATE neural_need_patterns
             SET status = 'proposal_generating', updated_at = NOW()
             WHERE id = $1`,
            [pattern.id]
          );

        } catch (error) {
          result.errors.push(`Error processing pattern ${pattern.id}: ${error}`);
        }
      }

      return result;

    } finally {
      client.release();
    }
  }

  function generateEvidenceSummary(evidence: any[]): PatternWithEvidence['evidenceSummary'] {
    const byType: Record<string, number> = {};
    const byUser: Record<string, number> = {};
    const timeline: Array<{ date: string; count: number; score: number }> = [];

    const dateMap: Record<string, { count: number; score: number }> = {};

    for (const e of evidence) {
      // By type
      byType[e.evidence_type] = (byType[e.evidence_type] || 0) + 1;

      // By user
      if (e.user_id) {
        byUser[e.user_id] = (byUser[e.user_id] || 0) + 1;
      }

      // Timeline
      const date = new Date(e.occurred_at).toISOString().split('T')[0];
      if (!dateMap[date]) {
        dateMap[date] = { count: 0, score: 0 };
```

```
    }
    dateMap[date].count++;
    dateMap[date].score += parseFloat(e.evidence_weight);
  }

  // Convert timeline map to array
  for (const [date, data] of Object.entries(dateMap)) {
    timeline.push({ date, ...data });
  }
  timeline.sort((a, b) => a.date.localeCompare(b.date));

  return { byType, byUser, timeline };
}

// ============================================================================
// Main Handler
// ============================================================================

export const handler: ScheduledHandler = async (event) => {
  console.log('Pattern analysis triggered:', event);

  const client = await pool.connect();

  try {
    // Get all active tenants
    const tenants = await client.query(
      `SELECT id FROM tenants WHERE status = 'active'`
    );

    const results: Record<string, AnalysisResult> = {};

    for (const tenant of tenants.rows) {
      results[tenant.id] = await analyzePatterns(tenant.id);
    }

    console.log('Pattern analysis complete:', results);

    return {
      statusCode: 200,
      body: JSON.stringify(results),
    };

  } finally {
    client.release();
  }
};
```

39.7 Proposal Generation Lambda (SQS Triggered)

```
// ============================================================================
// packages/lambdas/src/workflow-proposals/proposal-generator.ts
// SQS-triggered Lambda that generates workflow proposals using Neural Engine
// ============================================================================

import { SQSHandler, SQSRecord } from 'aws-lambda';
import { Pool } from 'pg';
import {
  PatternWithEvidence,
  WorkflowProposal,
  ProposedWorkflowDAG,
  NeuralProposalGenerationContext,
  NeuralProposalGenerationResult,
  WorkflowComplexity,
} from '@radiant/core/types/workflow-proposals';
import {
  COMPLEXITY_NODE_THRESHOLDS,
  PROPOSAL_CODE_PREFIX,
  MAX_PROPOSED_WORKFLOW_NODES,
} from '@radiant/core/constants/workflow-proposals';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  max: 10,
});

// ============================================================================
// Neural Engine Integration
// ============================================================================

interface NeuralEngineClient {
  generateProposal(context: NeuralProposalGenerationContext): Promise<NeuralProposalGenerati
}

async function createNeuralEngineClient(): Promise<NeuralEngineClient> {
  // In production, this would be an HTTP client to the Neural Engine service
  return {
    async generateProposal(context: NeuralProposalGenerationContext): Promise<NeuralProposal
      // This would call: POST /api/v2/internal/neural/generate-proposal
      // For now, implement intelligent proposal generation logic

      return generateIntelligentProposal(context);
```

```
    },
  };
}

async function generateIntelligentProposal(
  context: NeuralProposalGenerationContext
): Promise<NeuralProposalGenerationResult> {
  const { pattern, evidence, existingWorkflows, userSegmentProfile, tenantConstraints } = co

  // Analyze evidence to determine what kind of workflow is needed
  const evidenceAnalysis = analyzeEvidence(evidence);

  // Check if we have enough signal to propose
  if (evidenceAnalysis.confidence < 0.5) {
    return {
      success: false,
      confidence: evidenceAnalysis.confidence,
      reasoning: 'Insufficient evidence signal to generate confident proposal',
      rejectionReason: 'Low evidence confidence',
    };
  }

  // Determine workflow structure based on intent and evidence
  const workflowStructure = determineWorkflowStructure(
    pattern,
    evidenceAnalysis,
    userSegmentProfile,
    tenantConstraints
  );

  // Generate the DAG
  const proposedDAG = generateWorkflowDAG(
    workflowStructure,
    pattern,
    tenantConstraints
  );

  // Calculate confidence based on coverage
  const coverageEstimate = calculateCoverageEstimate(proposedDAG, evidence);
  const overallConfidence = (evidenceAnalysis.confidence + coverageEstimate) / 2;

  if (overallConfidence < 0.6) {
    return {
      success: false,
      confidence: overallConfidence,
      reasoning: 'Generated workflow does not sufficiently address the identified need',
```

```
      rejectionReason: 'Low coverage confidence',
    };
  }

  // Estimate quality improvement
  const qualityImprovement = estimateQualityImprovement(
    proposedDAG,
    existingWorkflows,
    evidenceAnalysis
  );

  return {
    success: true,
    proposal: {
      proposalCode: '', // Will be generated
      proposalName: generateProposalName(pattern),
      proposalDescription: generateProposalDescription(pattern, evidenceAnalysis),
      sourcePatternId: pattern.id,
      proposedWorkflow: proposedDAG,
      workflowCategory: pattern.patternSignature.intentCategory,
      workflowType: 'production_workflow',
      estimatedComplexity: determineComplexity(proposedDAG),
      neuralConfidence: overallConfidence,
      neuralReasoning: generateNeuralReasoning(pattern, evidenceAnalysis, proposedDAG),
      estimatedQualityImprovement: qualityImprovement,
      estimatedCoveragePercentage: coverageEstimate,
      evidenceCount: evidence.length,
      uniqueUsersAffected: pattern.uniqueUsersAffected,
      totalEvidenceScore: pattern.totalEvidenceScore,
      evidenceTimeSpanDays: calculateTimeSpan(pattern),
      evidenceSummary: {
        byType: pattern.evidenceSummary?.byType || {},
        topFailureReasons: extractTopFailureReasons(evidence),
        affectedDomains: pattern.patternSignature.domainHints,
      },
      adminStatus: 'pending_brain',
    },
    confidence: overallConfidence,
    reasoning: generateNeuralReasoning(pattern, evidenceAnalysis, proposedDAG),
    alternativeApproaches: generateAlternativeApproaches(pattern, evidenceAnalysis),
  };
}

// ============================================================================
// Evidence Analysis
// ============================================================================
```

```typescript
interface EvidenceAnalysis {
  confidence: number;
  primaryFailureType: string;
  failureReasons: string[];
  requiredCapabilities: string[];
  suggestedNodeTypes: string[];
  complexitySignal: WorkflowComplexity;
}

function analyzeEvidence(evidence: any[]): EvidenceAnalysis {
  const failureReasons: Record<string, number> = {};
  const capabilitySignals: Set<string> = new Set();
  let totalWeight = 0;

  for (const e of evidence) {
    totalWeight += parseFloat(e.evidence_weight);

    if (e.failure_reason) {
      failureReasons[e.failure_reason] = (failureReasons[e.failure_reason] || 0) + 1;

      // Extract capability signals from failure reasons
      const capabilities = extractCapabilities(e.failure_reason);
      capabilities.forEach(c => capabilitySignals.add(c));
    }

    if (e.user_feedback) {
      const capabilities = extractCapabilities(e.user_feedback);
      capabilities.forEach(c => capabilitySignals.add(c));
    }
  }

  // Sort failure reasons by frequency
  const sortedReasons = Object.entries(failureReasons)
    .sort((a, b) => b[1] - a[1])
    .map(([reason]) => reason);

  // Determine complexity from evidence volume and variety
  const complexitySignal = evidence.length > 20 ? 'complex' :
    evidence.length > 10 ? 'moderate' : 'simple';

  // Calculate confidence based on evidence consistency
  const reasonConsistency = sortedReasons.length > 0
    ? failureReasons[sortedReasons[0]] / evidence.length
    : 0;
  const confidence = Math.min(0.95, 0.3 + (totalWeight * 0.3) + (reasonConsistency * 0.4));
```

```
  return {
    confidence,
    primaryFailureType: sortedReasons[0] || 'unspecified',
    failureReasons: sortedReasons.slice(0, 5),
    requiredCapabilities: Array.from(capabilitySignals),
    suggestedNodeTypes: mapCapabilitiesToNodeTypes(Array.from(capabilitySignals)),
    complexitySignal: complexitySignal as WorkflowComplexity,
  };
}

function extractCapabilities(text: string): string[] {
  const capabilityPatterns: Record<string, string[]> = {
    'verification': ['wrong', 'incorrect', 'inaccurate', 'error', 'mistake', 'fact-check'],
    'multi_source': ['more sources', 'compare', 'multiple', 'different perspectives'],
    'depth': ['more detail', 'deeper', 'comprehensive', 'thorough', 'in-depth'],
    'reasoning': ['explain', 'reasoning', 'logic', 'step-by-step', 'how'],
    'creativity': ['creative', 'original', 'unique', 'novel', 'innovative'],
    'structure': ['organize', 'structure', 'format', 'outline', 'layout'],
    'iteration': ['refine', 'improve', 'iterate', 'better', 'enhance'],
  };

  const found: string[] = [];
  const lowerText = text.toLowerCase();

  for (const [capability, patterns] of Object.entries(capabilityPatterns)) {
    if (patterns.some(p => lowerText.includes(p))) {
      found.push(capability);
    }
  }

  return found;
}

function mapCapabilitiesToNodeTypes(capabilities: string[]): string[] {
  const mapping: Record<string, string[]> = {
    'verification': ['fact_check', 'cove_verification', 'multi_source_verify'],
    'multi_source': ['parallel_search', 'source_aggregation', 'perspective_synthesis'],
    'depth': ['recursive_elaboration', 'detail_expansion', 'exhaustive_analysis'],
    'reasoning': ['chain_of_thought', 'step_by_step', 'reasoning_chain'],
    'creativity': ['creative_expansion', 'brainstorm', 'divergent_thinking'],
    'structure': ['outline_generator', 'structure_organizer', 'format_optimizer'],
    'iteration': ['quality_loop', 'refinement_cycle', 'iterative_improvement'],
  };

  const nodeTypes: Set<string> = new Set();
```

```
  for (const capability of capabilities) {
    const types = mapping[capability] || [];
    types.forEach(t => nodeTypes.add(t));
  }

  return Array.from(nodeTypes);
}

// ============================================================================
// Workflow Structure Determination
// ============================================================================

interface WorkflowStructure {
  entryStrategy: 'single' | 'parallel' | 'conditional';
  coreNodeTypes: string[];
  verificationRequired: boolean;
  iterationRequired: boolean;
  exitStrategy: 'single' | 'merge' | 'select_best';
}

function determineWorkflowStructure(
  pattern: PatternWithEvidence,
  analysis: EvidenceAnalysis,
  userProfile: NeuralProposalGenerationContext['userSegmentProfile'],
  constraints: NeuralProposalGenerationContext['tenantConstraints']
): WorkflowStructure {
  // Base structure on intent category
  const intentCategory = pattern.patternSignature.intentCategory;

  let structure: WorkflowStructure = {
    entryStrategy: 'single',
    coreNodeTypes: [],
    verificationRequired: false,
    iterationRequired: false,
    exitStrategy: 'single',
  };

  // Determine entry strategy
  if (analysis.requiredCapabilities.includes('multi_source')) {
    structure.entryStrategy = 'parallel';
    structure.exitStrategy = 'merge';
  }

  // Add core node types based on intent
  switch (intentCategory) {
```

```
    case 'research':
      structure.coreNodeTypes = ['web_search', 'source_aggregation', 'citation_formatter'];
      structure.verificationRequired = true;
      break;
    case 'analysis':
      structure.coreNodeTypes = ['data_extraction', 'analysis_engine', 'insight_generator'];
      structure.iterationRequired = true;
      break;
    case 'creation':
      structure.coreNodeTypes = ['outline_generator', 'content_generator', 'quality_reviewer
      structure.iterationRequired = true;
      break;
    case 'verification':
      structure.coreNodeTypes = ['claim_extractor', 'fact_checker', 'confidence_scorer'];
      structure.entryStrategy = 'parallel';
      structure.exitStrategy = 'merge';
      break;
    default:
      structure.coreNodeTypes = ['llm_processor', 'response_formatter'];
  }

  // Add suggested node types from evidence
  structure.coreNodeTypes = [
    ...structure.coreNodeTypes,
    ...analysis.suggestedNodeTypes.filter(t => !structure.coreNodeTypes.includes(t)),
  ];

  // Limit to max allowed nodes
  if (structure.coreNodeTypes.length > constraints.maxWorkflowNodes - 2) {
    structure.coreNodeTypes = structure.coreNodeTypes.slice(0, constraints.maxWorkflowNodes
  }

  // Add verification if confidence is low
  if (analysis.confidence < 0.7 && !structure.verificationRequired) {
    structure.verificationRequired = true;
  }

  return structure;
}

// ============================================================================
// DAG Generation
// ============================================================================

function generateWorkflowDAG(
  structure: WorkflowStructure,
```

```
  pattern: PatternWithEvidence,
  constraints: NeuralProposalGenerationContext['tenantConstraints']
): ProposedWorkflowDAG {
  const nodes: ProposedWorkflowDAG['nodes'] = [];
  const edges: ProposedWorkflowDAG['edges'] = [];
  let nodeIdCounter = 0;
  let yPosition = 100;

  const generateNodeId = () => `node_${++nodeIdCounter}`;

  // Entry node
  const entryNode = {
    id: generateNodeId(),
    type: 'input',
    name: 'Input',
    config: {},
    position: { x: 100, y: yPosition },
    connections: [],
  };
  nodes.push(entryNode);
  let previousNodeIds = [entryNode.id];

  yPosition += 150;

  // Handle entry strategy
  if (structure.entryStrategy === 'parallel') {
    const parallelNodes: string[] = [];
    const xPositions = [-150, 0, 150];

    for (let i = 0; i < Math.min(3, structure.coreNodeTypes.length); i++) {
      const node = {
        id: generateNodeId(),
        type: structure.coreNodeTypes[i] || 'processor',
        name: formatNodeName(structure.coreNodeTypes[i] || 'Processor'),
        config: {},
        position: { x: 100 + xPositions[i], y: yPosition },
        connections: [],
      };
      nodes.push(node);
      parallelNodes.push(node.id);

      edges.push({ from: entryNode.id, to: node.id });
    }

    previousNodeIds = parallelNodes;
    yPosition += 150;
```

```
    // Merge node if parallel
    if (structure.exitStrategy === 'merge') {
      const mergeNode = {
        id: generateNodeId(),
        type: 'merge',
        name: 'Merge Results',
        config: { strategy: 'combine' },
        position: { x: 100, y: yPosition },
        connections: [],
      };
      nodes.push(mergeNode);

      for (const prevId of parallelNodes) {
        edges.push({ from: prevId, to: mergeNode.id });
      }

      previousNodeIds = [mergeNode.id];
      yPosition += 150;
    }

    // Add remaining core nodes sequentially
    for (let i = 3; i < structure.coreNodeTypes.length; i++) {
      const node = {
        id: generateNodeId(),
        type: structure.coreNodeTypes[i],
        name: formatNodeName(structure.coreNodeTypes[i]),
        config: {},
        position: { x: 100, y: yPosition },
        connections: [],
      };
      nodes.push(node);

      for (const prevId of previousNodeIds) {
        edges.push({ from: prevId, to: node.id });
      }

      previousNodeIds = [node.id];
      yPosition += 150;
    }

  } else {
    // Sequential entry
    for (const nodeType of structure.coreNodeTypes) {
      const node = {
        id: generateNodeId(),
```

```
      type: nodeType,
      name: formatNodeName(nodeType),
      config: {},
      position: { x: 100, y: yPosition },
      connections: [],
    };
    nodes.push(node);

    for (const prevId of previousNodeIds) {
      edges.push({ from: prevId, to: node.id });
    }

    previousNodeIds = [node.id];
    yPosition += 150;
  }
}

// Add verification node if required
if (structure.verificationRequired) {
  const verifyNode = {
    id: generateNodeId(),
    type: 'verification',
    name: 'Quality Verification',
    config: { minConfidence: 0.7 },
    position: { x: 100, y: yPosition },
    connections: [],
  };
  nodes.push(verifyNode);

  for (const prevId of previousNodeIds) {
    edges.push({ from: prevId, to: verifyNode.id });
  }

  previousNodeIds = [verifyNode.id];
  yPosition += 150;
}

// Add iteration loop if required
if (structure.iterationRequired) {
  const qualityCheckNode = {
    id: generateNodeId(),
    type: 'quality_check',
    name: 'Quality Check',
    config: { threshold: 0.8, maxIterations: 3 },
    position: { x: 100, y: yPosition },
    connections: [],
```

```
    };
    nodes.push(qualityCheckNode);

    for (const prevId of previousNodeIds) {
      edges.push({ from: prevId, to: qualityCheckNode.id });
    }

    // Add edge back to refinement (simplified - would need proper loop handling)
    previousNodeIds = [qualityCheckNode.id];
    yPosition += 150;
  }

  // Output node
  const outputNode = {
    id: generateNodeId(),
    type: 'output',
    name: 'Output',
    config: {},
    position: { x: 100, y: yPosition },
    connections: [],
  };
  nodes.push(outputNode);

  for (const prevId of previousNodeIds) {
    edges.push({ from: prevId, to: outputNode.id });
  }

  return {
    nodes,
    edges,
    entryPoint: entryNode.id,
    exitPoints: [outputNode.id],
    metadata: {
      estimatedLatencyMs: estimateLatency(nodes),
      estimatedCostPer1k: estimateCost(nodes),
      requiredModels: extractRequiredModels(nodes),
      requiredServices: extractRequiredServices(nodes),
    },
  };
}

function formatNodeName(nodeType: string): string {
  return nodeType
    .split('_')
    .map(word => word.charAt(0).toUpperCase() + word.slice(1))
    .join(' ');
```

```
}

function estimateLatency(nodes: ProposedWorkflowDAG['nodes']): number {
  // Rough estimate: 500ms base + 200ms per node
  return 500 + (nodes.length * 200);
}

function estimateCost(nodes: ProposedWorkflowDAG['nodes']): number {
  // Rough estimate: $0.01 base + $0.005 per node per 1k requests
  return 0.01 + (nodes.length * 0.005);
}

function extractRequiredModels(nodes: ProposedWorkflowDAG['nodes']): string[] {
  const modelTypes: Record<string, string> = {
    'llm_processor': 'claude-3-5-sonnet',
    'content_generator': 'claude-3-5-sonnet',
    'fact_checker': 'gpt-4-turbo',
    'analysis_engine': 'claude-3-5-sonnet',
  };

  const models: Set<string> = new Set();
  for (const node of nodes) {
    if (modelTypes[node.type]) {
      models.add(modelTypes[node.type]);
    }
  }

  return Array.from(models);
}

function extractRequiredServices(nodes: ProposedWorkflowDAG['nodes']): string[] {
  const serviceTypes: Record<string, string> = {
    'web_search': 'search_service',
    'source_aggregation': 'aggregation_service',
    'fact_checker': 'verification_service',
  };

  const services: Set<string> = new Set();
  for (const node of nodes) {
    if (serviceTypes[node.type]) {
      services.add(serviceTypes[node.type]);
    }
  }

  return Array.from(services);
}
```

```typescript
// ============================================================================
// Helper Functions
// ============================================================================

function calculateCoverageEstimate(
  dag: ProposedWorkflowDAG,
  evidence: any[]
): number {
  // Estimate what percentage of evidence cases this workflow would address
  // Based on node types matching required capabilities

  const capabilities: Set<string> = new Set();
  for (const e of evidence) {
    if (e.failure_reason) {
      extractCapabilities(e.failure_reason).forEach(c => capabilities.add(c));
    }
  }

  const nodeTypes = new Set(dag.nodes.map(n => n.type));
  const capabilityMapping = mapCapabilitiesToNodeTypes(Array.from(capabilities));

  let covered = 0;
  for (const nodeType of capabilityMapping) {
    if (nodeTypes.has(nodeType)) {
      covered++;
    }
  }

  return capabilityMapping.length > 0
    ? Math.min(0.95, covered / capabilityMapping.length)
    : 0.6;
}

function estimateQualityImprovement(
  dag: ProposedWorkflowDAG,
  existingWorkflows: NeuralProposalGenerationContext['existingWorkflows'],
  analysis: EvidenceAnalysis
): number {
  // Estimate quality improvement over existing workflows
  // Based on failure rate of existing and new capabilities

  const avgFailureRate = existingWorkflows.length > 0
    ? existingWorkflows.reduce((sum, w) => sum + w.failureRate, 0) / existingWorkflows.lengt
    : 0.5;
```

```
  const newCapabilities = analysis.suggestedNodeTypes.filter(
    t => !existingWorkflows.some(w => w.name.toLowerCase().includes(t))
  );

  const capabilityBonus = newCapabilities.length * 0.05;

  return Math.min(0.40, avgFailureRate * 0.5 + capabilityBonus);
}

function generateProposalName(pattern: PatternWithEvidence): string {
  const intent = pattern.patternSignature.intentCategory;
  const domains = pattern.patternSignature.domainHints;

  const domainPrefix = domains.length > 0 ? `${domains[0].charAt(0).toUpperCase()}${domains
  const intentName = intent.charAt(0).toUpperCase() + intent.slice(1);

  return `${domainPrefix}${intentName} Enhancement Workflow`;
}

function generateProposalDescription(
  pattern: PatternWithEvidence,
  analysis: EvidenceAnalysis
): string {
  const reasons = analysis.failureReasons.slice(0, 3).join(', ');
  const users = pattern.uniqueUsersAffected;
  const evidence = pattern.evidenceCount;

  return `This workflow addresses user needs identified through ${evidence} evidence signals
}

function generateNeuralReasoning(
  pattern: PatternWithEvidence,
  analysis: EvidenceAnalysis,
  dag: ProposedWorkflowDAG
): string {
  return `Based on analysis of ${pattern.evidenceCount} evidence records with ${(analysis.co
1. Primary failure pattern: ${analysis.primaryFailureType}
2. Required capabilities: ${analysis.requiredCapabilities.join(', ')}
3. Proposed solution: ${dag.nodes.length}-node workflow with ${dag.metadata.requiredModels.j
4. Estimated coverage: ${(calculateCoverageEstimate(dag, pattern.evidence || []) * 100).toFi
}

function generateAlternativeApproaches(
  pattern: PatternWithEvidence,
  analysis: EvidenceAnalysis
): NeuralProposalGenerationResult['alternativeApproaches'] {
```

```
  return [
    {
      description: 'Simplified single-model approach with enhanced prompting',
      confidence: analysis.confidence * 0.7,
      tradeoffs: ['Lower latency', 'Lower cost', 'Potentially lower quality'],
    },
    {
      description: 'Extended multi-model ensemble with voting',
      confidence: analysis.confidence * 1.1,
      tradeoffs: ['Higher quality', 'Higher latency', 'Higher cost'],
    },
  ];
}

function determineComplexity(dag: ProposedWorkflowDAG): WorkflowComplexity {
  const nodeCount = dag.nodes.length;

  for (const [complexity, thresholds] of Object.entries(COMPLEXITY_NODE_THRESHOLDS)) {
    if (nodeCount >= thresholds.min && nodeCount <= thresholds.max) {
      return complexity as WorkflowComplexity;
    }
  }

  return 'moderate';
}

function calculateTimeSpan(pattern: PatternWithEvidence): number {
  if (!pattern.firstOccurrence || !pattern.lastOccurrence) return 0;

  const first = new Date(pattern.firstOccurrence).getTime();
  const last = new Date(pattern.lastOccurrence).getTime();

  return Math.ceil((last - first) / (1000 * 60 * 60 * 24));
}

function extractTopFailureReasons(evidence: any[]): string[] {
  const reasons: Record<string, number> = {};

  for (const e of evidence) {
    if (e.failure_reason) {
      reasons[e.failure_reason] = (reasons[e.failure_reason] || 0) + 1;
    }
  }

  return Object.entries(reasons)
    .sort((a, b) => b[1] - a[1])
```

```
    .slice(0, 5)
    .map(([reason]) => reason);
}

// ============================================================================
// Proposal Code Generation
// ============================================================================

async function generateProposalCode(client: Pool, tenantId: string): Promise<string> {
  const year = new Date().getFullYear();

  const result = await client.query(
    `SELECT COUNT(*) + 1 as seq
     FROM workflow_proposals
     WHERE tenant_id = $1
       AND EXTRACT(YEAR FROM created_at) = $2`,
    [tenantId, year]
  );

  const sequence = String(result.rows[0].seq).padStart(3, '0');
  return `${PROPOSAL_CODE_PREFIX}-${year}-${sequence}`;
}

// ============================================================================
// Main Handler
// ============================================================================

export const handler: SQSHandler = async (event) => {
  const neuralEngine = await createNeuralEngineClient();
  const client = await pool.connect();

  try {
    for (const record of event.Records) {
      await processRecord(record, neuralEngine, client);
    }
  } finally {
    client.release();
  }
};

async function processRecord(
  record: SQSRecord,
  neuralEngine: NeuralEngineClient,
  client: Pool
): Promise<void> {
  const message = JSON.parse(record.body);
```

```
if (message.type !== 'generate_proposal') {
  console.log('Unknown message type:', message.type);
  return;
}

const { tenantId, pattern } = message as {
  tenantId: string;
  pattern: PatternWithEvidence;
};

console.log(`Generating proposal for pattern ${pattern.id} in tenant ${tenantId}`);

await client.query(`SET app.current_tenant_id = '${tenantId}'`);

try {
  // Build context for Neural Engine
  const existingWorkflows = await client.query(
    `SELECT id, name,
      (SELECT COUNT(*) FROM neural_need_evidence WHERE attempted_workflow_id = w.id) as fa
      (SELECT COUNT(*) FROM execution_manifests WHERE workflow_id = w.id) as total_count
     FROM production_workflows w
     WHERE tenant_id = $1
     LIMIT 50`,
    [tenantId]
  );

  const thresholdConfig = await client.query(
    `SELECT * FROM proposal_threshold_config WHERE tenant_id = $1`,
    [tenantId]
  );

  const context: NeuralProposalGenerationContext = {
    pattern,
    evidence: pattern.evidence || [],
    existingWorkflows: existingWorkflows.rows.map(w => ({
      id: w.id,
      name: w.name,
      similarity: 0.5, // Would be calculated from embeddings
      failureRate: w.total_count > 0 ? w.failure_count / w.total_count : 0,
    })),
    userSegmentProfile: {
      commonIntents: [pattern.patternSignature.intentCategory],
      preferredComplexity: 'moderate',
      domainDistribution: Object.fromEntries(
        pattern.patternSignature.domainHints.map(d => [d, 1])
```

```
    ),
  },
  tenantConstraints: {
    maxWorkflowNodes: MAX_PROPOSED_WORKFLOW_NODES,
    allowedNodeTypes: [], // All types allowed by default
    requiredComplianceChecks: [],
  },
};

// Generate proposal
const result = await neuralEngine.generateProposal(context);

if (!result.success || !result.proposal) {
  console.log(`Proposal generation failed for pattern ${pattern.id}: ${result.rejectionR

  // Update pattern status
  await client.query(
    `UPDATE neural_need_patterns
     SET status = 'accumulating',
         confidence_threshold_met = FALSE,
         updated_at = NOW()
     WHERE id = $1`,
    [pattern.id]
  );

  return;
}

// Generate proposal code
const proposalCode = await generateProposalCode(client, tenantId);

// Insert proposal
const proposalResult = await client.query<{ id: string }>(
  `INSERT INTO workflow_proposals (
    tenant_id, proposal_code, proposal_name, proposal_description,
    source_pattern_id, proposed_workflow, workflow_category, workflow_type,
    estimated_complexity, neural_confidence, neural_reasoning,
    estimated_quality_improvement, estimated_coverage_percentage,
    evidence_count, unique_users_affected, total_evidence_score,
    evidence_time_span_days, evidence_summary, admin_status
  ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16, $17,
  RETURNING id`,
  [
    tenantId,
    proposalCode,
    result.proposal.proposalName,
```

```
          result.proposal.proposalDescription,
          pattern.id,
          JSON.stringify(result.proposal.proposedWorkflow),
          result.proposal.workflowCategory,
          result.proposal.workflowType,
          result.proposal.estimatedComplexity,
          result.confidence,
          result.reasoning,
          result.proposal.estimatedQualityImprovement,
          result.proposal.estimatedCoveragePercentage,
          result.proposal.evidenceCount,
          result.proposal.uniqueUsersAffected,
          result.proposal.totalEvidenceScore,
          result.proposal.evidenceTimeSpanDays,
          JSON.stringify(result.proposal.evidenceSummary),
          'pending_brain',
        ]
      );

      // Update pattern
      await client.query(
        `UPDATE neural_need_patterns
         SET status = 'proposal_generated',
             confidence_threshold_met = TRUE,
             proposal_id = $2,
             updated_at = NOW()
         WHERE id = $1`,
        [pattern.id, proposalResult.rows[0].id]
      );

      console.log(`Created proposal ${proposalCode} for pattern ${pattern.id}`);

    } catch (error) {
      console.error(`Error generating proposal for pattern ${pattern.id}:`, error);

      // Reset pattern status on error
      await client.query(
        `UPDATE neural_need_patterns
         SET status = 'threshold_met', updated_at = NOW()
         WHERE id = $1`,
        [pattern.id]
      );

      throw error;
    }
  }
```

This chunk contains: - Evidence Collection Lambda (pattern signature generation, keyword extraction, intent classification) - Pattern Analysis Lambda (scheduled job that finds qualified patterns) - Proposal Generation Lambda (SQS-triggered, Neural Engine integration, DAG generation)

Next chunk will contain Brain Governor Review, Admin APIs, and React Admin Dashboard components.

39.8 Brain Governor Review Lambda (SQS Triggered)

```
// ============================================================================
// packages/lambdas/src/workflow-proposals/brain-reviewer.ts
// Brain Governor reviews proposals before they reach admin queue
// ============================================================================

import { SQSHandler, SQSRecord } from 'aws-lambda';
import { Pool } from 'pg';
import { SQSClient, SendMessageCommand } from '@aws-sdk/client-sqs';
import {
  WorkflowProposal,
  BrainProposalReviewContext,
  BrainProposalReviewResult,
  BrainRiskAssessment,
  ProposalThresholdConfig,
} from '@radiant/core/types/workflow-proposals';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  max: 10,
});

const sqs = new SQSClient({});
const ADMIN_NOTIFICATION_QUEUE_URL = process.env.ADMIN_NOTIFICATION_QUEUE_URL!;

// ============================================================================
// Risk Assessment
// ============================================================================

function assessCostRisk(proposal: WorkflowProposal): number {
  const estimatedCost = proposal.proposedWorkflow.metadata.estimatedCostPer1k;
  const nodeCount = proposal.proposedWorkflow.nodes.length;
  const modelCount = proposal.proposedWorkflow.metadata.requiredModels.length;
```

```
  // Higher cost = higher risk
  let risk = 0;

  if (estimatedCost > 0.05) risk += 0.2;
  if (estimatedCost > 0.10) risk += 0.2;
  if (estimatedCost > 0.20) risk += 0.2;

  if (nodeCount > 10) risk += 0.1;
  if (nodeCount > 15) risk += 0.1;

  if (modelCount > 3) risk += 0.1;

  return Math.min(1.0, risk);
}

function assessLatencyRisk(proposal: WorkflowProposal): number {
  const estimatedLatency = proposal.proposedWorkflow.metadata.estimatedLatencyMs;
  const nodeCount = proposal.proposedWorkflow.nodes.length;
  const hasParallelNodes = proposal.proposedWorkflow.edges.some(
    (edge, _, edges) => edges.filter(e => e.from === edge.from).length > 1
  );

  let risk = 0;

  if (estimatedLatency > 2000) risk += 0.2;
  if (estimatedLatency > 5000) risk += 0.2;
  if (estimatedLatency > 10000) risk += 0.3;

  // Parallel execution can help or hurt
  if (!hasParallelNodes && nodeCount > 5) risk += 0.2;

  return Math.min(1.0, risk);
}

function assessQualityRisk(proposal: WorkflowProposal): number {
  const hasVerification = proposal.proposedWorkflow.nodes.some(
    n => n.type.includes('verification') || n.type.includes('quality')
  );
  const confidence = proposal.neuralConfidence;
  const coverage = proposal.estimatedCoveragePercentage || 0;

  let risk = 0;

  // Low confidence = high risk
  if (confidence < 0.80) risk += 0.2;
  if (confidence < 0.70) risk += 0.2;
```

```
  if (confidence < 0.60) risk += 0.2;

  // Low coverage = high risk
  if (coverage < 0.70) risk += 0.1;
  if (coverage < 0.60) risk += 0.1;

  // No verification = higher risk
  if (!hasVerification) risk += 0.1;

  return Math.min(1.0, risk);
}

function assessComplianceRisk(proposal: WorkflowProposal): number {
  // Check for nodes that might have compliance implications
  const riskyNodeTypes = ['external_api', 'data_export', 'pii_handler', 'external_search'];

  let risk = 0;

  for (const node of proposal.proposedWorkflow.nodes) {
    if (riskyNodeTypes.some(t => node.type.includes(t))) {
      risk += 0.15;
    }
  }

  // Check if workflow category has compliance requirements
  const highComplianceCategories = ['medical', 'legal', 'financial'];
  if (highComplianceCategories.some(c => proposal.workflowCategory?.includes(c))) {
    risk += 0.2;
  }

  return Math.min(1.0, risk);
}

function performRiskAssessment(proposal: WorkflowProposal): BrainRiskAssessment {
  const costRisk = assessCostRisk(proposal);
  const latencyRisk = assessLatencyRisk(proposal);
  const qualityRisk = assessQualityRisk(proposal);
  const complianceRisk = assessComplianceRisk(proposal);

  // Overall risk is weighted average
  const overallRisk = (
    costRisk * 0.25 +
    latencyRisk * 0.25 +
    qualityRisk * 0.30 +
    complianceRisk * 0.20
  );
```

```
  // Identify risk factors
  const riskFactors: string[] = [];
  if (costRisk > 0.3) riskFactors.push('High estimated cost');
  if (latencyRisk > 0.4) riskFactors.push('High estimated latency');
  if (qualityRisk > 0.3) riskFactors.push('Quality concerns');
  if (complianceRisk > 0.2) riskFactors.push('Compliance considerations');

  // Suggest mitigations
  const mitigations: string[] = [];
  if (costRisk > 0.3) mitigations.push('Consider model alternatives or caching');
  if (latencyRisk > 0.4) mitigations.push('Add parallel execution or reduce nodes');
  if (qualityRisk > 0.3) mitigations.push('Add verification nodes');
  if (complianceRisk > 0.2) mitigations.push('Add compliance checks or audit logging');

  return {
    costRisk,
    latencyRisk,
    qualityRisk,
    complianceRisk,
    overallRisk,
    riskFactors,
    mitigations,
  };
}

// ============================================================================
// Brain Review Logic
// ============================================================================

async function reviewProposal(
  context: BrainProposalReviewContext
): Promise<BrainProposalReviewResult> {
  const { proposal, pattern, tenantConfig, recentProposalCount, similarExistingWorkflows } =

  // Perform risk assessment
  const riskAssessment = performRiskAssessment(proposal);

  // Check against thresholds
  const maxCostRisk = tenantConfig.maxCostRisk || 0.30;
  const maxLatencyRisk = tenantConfig.maxLatencyRisk || 0.40;
  const minQualityConfidence = tenantConfig.minQualityConfidence || 0.70;
  const maxComplianceRisk = tenantConfig.maxComplianceRisk || 0.10;

  // Determine if we should veto
  let vetoReason: string | undefined;
```

```
if (riskAssessment.costRisk > maxCostRisk) {
  vetoReason = `Cost risk (${(riskAssessment.costRisk * 100).toFixed(0)}%) exceeds thresho
} else if (riskAssessment.latencyRisk > maxLatencyRisk) {
  vetoReason = `Latency risk (${(riskAssessment.latencyRisk * 100).toFixed(0)}%) exceeds t
} else if (proposal.neuralConfidence < minQualityConfidence) {
  vetoReason = `Neural confidence (${(proposal.neuralConfidence * 100).toFixed(0)}%) belov
} else if (riskAssessment.complianceRisk > maxComplianceRisk) {
  vetoReason = `Compliance risk (${(riskAssessment.complianceRisk * 100).toFixed(0)}%) exc
}

// Check for duplicate/similar workflows
const highSimilarityWorkflow = similarExistingWorkflows.find(w => w.similarity > 0.85);
if (highSimilarityWorkflow) {
  vetoReason = `Very similar workflow already exists: ${highSimilarityWorkflow.name} (${(h
}

// Check rate limits (redundant check, but important)
if (recentProposalCount.today >= (tenantConfig.maxProposalsPerDay || 10)) {
  vetoReason = 'Daily proposal limit reached';
} else if (recentProposalCount.thisWeek >= (tenantConfig.maxProposalsPerWeek || 30)) {
  vetoReason = 'Weekly proposal limit reached';
}

// Determine priority for admin review
let adminPriority: 'low' | 'medium' | 'high' | 'urgent' = 'medium';

if (proposal.uniqueUsersAffected > 50) adminPriority = 'high';
if (proposal.uniqueUsersAffected > 100) adminPriority = 'urgent';
if (proposal.totalEvidenceScore > 5.0) adminPriority = 'high';
if (riskAssessment.overallRisk < 0.2 && proposal.neuralConfidence > 0.85) adminPriority =

// Generate reasoning
const reasoning = generateReviewReasoning(proposal, riskAssessment, vetoReason);

// Check if modifications are suggested
let suggestedModifications: BrainProposalReviewResult['suggestedModifications'];

if (!vetoReason && riskAssessment.overallRisk > 0.25) {
  suggestedModifications = generateSuggestedModifications(proposal, riskAssessment);
}

return {
  approved: !vetoReason,
  riskAssessment,
  reasoning,
```

```
    vetoReason,
    suggestedModifications,
    escalateToAdmin: !vetoReason,
    adminPriority: vetoReason ? undefined : adminPriority,
  };
}

function generateReviewReasoning(
  proposal: WorkflowProposal,
  risk: BrainRiskAssessment,
  vetoReason?: string
): string {
  if (vetoReason) {
    return `Proposal vetoed: ${vetoReason}. Risk assessment: Cost ${(risk.costRisk * 100).to

  return `Proposal approved for admin review. Overall risk: ${(risk.overallRisk * 100).toFi
}

function generateSuggestedModifications(
  proposal: WorkflowProposal,
  risk: BrainRiskAssessment
): Partial<WorkflowProposal['proposedWorkflow']> | undefined {
  const modifications: Partial<WorkflowProposal['proposedWorkflow']> = {};

  // If latency risk is high, suggest adding parallel execution
  if (risk.latencyRisk > 0.3) {
    // This would involve restructuring the DAG - simplified here
    modifications.metadata = {
      ...proposal.proposedWorkflow.metadata,
      // Add suggestion note
    };
  }

  // If quality risk is high and no verification, suggest adding it
  if (risk.qualityRisk > 0.3) {
    const hasVerification = proposal.proposedWorkflow.nodes.some(
      n => n.type.includes('verification')
    );

    if (!hasVerification) {
      // Suggest adding verification node
      modifications.nodes = [
        ...proposal.proposedWorkflow.nodes,
        {
          id: `suggested_verification_${Date.now()}`,
```

```
          type: 'quality_verification',
          name: 'Quality Verification (Suggested)',
          config: { minConfidence: 0.7 },
          position: { x: 100, y: 1000 },
          connections: [],
        },
      ];
    }
  }

  return Object.keys(modifications).length > 0 ? modifications : undefined;
}

// ============================================================================
// Main Handler
// ============================================================================

export const handler: SQSHandler = async (event) => {
  const client = await pool.connect();

  try {
    for (const record of event.Records) {
      await processRecord(record, client);
    }
  } finally {
    client.release();
  }
};

async function processRecord(record: SQSRecord, client: Pool): Promise<void> {
  const message = JSON.parse(record.body);

  if (message.type !== 'brain_review') {
    console.log('Unknown message type:', message.type);
    return;
  }

  const { tenantId, proposalId } = message;

  console.log(`Brain reviewing proposal ${proposalId} for tenant ${tenantId}`);

  await client.query(`SET app.current_tenant_id = '${tenantId}'`);

  try {
    // Fetch proposal with pattern
    const proposalResult = await client.query<WorkflowProposal>(
```

```
  `SELECT * FROM workflow_proposals WHERE id = $1`,
  [proposalId]
);

if (proposalResult.rows.length === 0) {
  console.error(`Proposal ${proposalId} not found`);
  return;
}

const proposal = proposalResult.rows[0];

// Skip if not pending brain review
if (proposal.admin_status !== 'pending_brain') {
  console.log(`Proposal ${proposalId} not pending brain review, skipping`);
  return;
}

// Fetch pattern
const patternResult = await client.query(
  `SELECT * FROM neural_need_patterns WHERE id = $1`,
  [proposal.source_pattern_id]
);

// Fetch config
const configResult = await client.query<ProposalThresholdConfig>(
  `SELECT * FROM proposal_threshold_config WHERE tenant_id = $1`,
  [tenantId]
);
const config = configResult.rows[0] || {};

// Fetch recent proposal counts
const countsResult = await client.query(
  `SELECT
    COUNT(*) FILTER (WHERE created_at > NOW() - INTERVAL '1 day') as today,
    COUNT(*) FILTER (WHERE created_at > NOW() - INTERVAL '7 days') as this_week
   FROM workflow_proposals WHERE tenant_id = $1`,
  [tenantId]
);

// Fetch similar workflows
const similarResult = await client.query(
  `SELECT id, name, 0.5 as similarity
   FROM production_workflows
   WHERE tenant_id = $1
     AND category = $2
   LIMIT 10`,
```

```
    [tenantId, proposal.workflow_category]
  );

  // Build context
  const context: BrainProposalReviewContext = {
    proposal,
    pattern: patternResult.rows[0],
    tenantConfig: config,
    currentWorkflowCount: 0,
    recentProposalCount: {
      today: parseInt(countsResult.rows[0].today),
      thisWeek: parseInt(countsResult.rows[0].this_week),
    },
    similarExistingWorkflows: similarResult.rows.map(w => ({
      id: w.id,
      name: w.name,
      similarity: w.similarity,
    })),
  };

  // Perform review
  const reviewResult = await reviewProposal(context);

  // Update proposal
  const newStatus = reviewResult.approved ? 'pending_admin' : 'declined';

  await client.query(
    `UPDATE workflow_proposals SET
      brain_approved = $2,
      brain_review_timestamp = NOW(),
      brain_risk_assessment = $3,
      brain_veto_reason = $4,
      brain_modifications = $5,
      admin_status = $6,
      updated_at = NOW()
    WHERE id = $1`,
    [
      proposalId,
      reviewResult.approved,
      JSON.stringify(reviewResult.riskAssessment),
      reviewResult.vetoReason,
      reviewResult.suggestedModifications ? JSON.stringify(reviewResult.suggestedModificat
      newStatus,
    ]
  );
```

```
    // Record review
    await client.query(
      `INSERT INTO workflow_proposal_reviews (
        proposal_id, reviewer_type, action, previous_status, new_status,
        review_notes, risk_assessment
      ) VALUES ($1, 'brain', $2, 'pending_brain', $3, $4, $5)`,
      [
        proposalId,
        reviewResult.approved ? 'approve' : 'decline',
        newStatus,
        reviewResult.reasoning,
        JSON.stringify(reviewResult.riskAssessment),
      ]
    );

    // Notify admins if approved
    if (reviewResult.approved) {
      await sqs.send(new SendMessageCommand({
        QueueUrl: ADMIN_NOTIFICATION_QUEUE_URL,
        MessageBody: JSON.stringify({
          type: 'new_proposal',
          tenantId,
          proposalId,
          proposalCode: proposal.proposal_code,
          priority: reviewResult.adminPriority,
          riskLevel: reviewResult.riskAssessment.overallRisk,
        }),
      }));
    }

    console.log(`Brain ${reviewResult.approved ? 'approved' : 'vetoed'} proposal ${proposall

  } catch (error) {
    console.error(`Error in brain review for proposal ${proposalId}:`, error);
    throw error;
  }
}
```

39.9 Admin API Endpoints

```
// ============================================================================
// packages/lambdas/src/workflow-proposals/admin-api.ts
// Admin API for proposal management
// ============================================================================
```

```typescript
import { APIGatewayProxyHandler, APIGatewayProxyResult } from 'aws-lambda';
import { Pool } from 'pg';
import {
  GetProposalsRequest,
  GetProposalsResponse,
  ReviewProposalRequest,
  ReviewProposalResponse,
  TestProposalRequest,
  TestProposalResponse,
  PublishProposalRequest,
  PublishProposalResponse,
  UpdateThresholdConfigRequest,
  UpdateEvidenceWeightsRequest,
  WorkflowProposal,
  ProposalReview,
} from '@radiant/core/types/workflow-proposals';
import { getTenantId, getUserId, createResponse, isAdmin, logAudit } from '../utils';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  max: 10,
});

// ============================================================================
// GET /api/v2/admin/proposals - List proposals
// ============================================================================

export const listProposals: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxyR
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    const params = event.queryStringParameters || {};
    const statusFilter = params.status?.split(',') || ['pending_admin'];
    const limit = Math.min(parseInt(params.limit || '20'), 100);
    const offset = parseInt(params.offset || '0');
```

```
    // Build query
    let query = `
      SELECT wp.*, nnp.pattern_name, nnp.detected_intent
      FROM workflow_proposals wp
      JOIN neural_need_patterns nnp ON nnp.id = wp.source_pattern_id
      WHERE wp.tenant_id = $1
    `;
    const queryParams: any[] = [tenantId];

    if (statusFilter.length > 0) {
      queryParams.push(statusFilter);
      query += ` AND wp.admin_status = ANY($${queryParams.length})`;
    }

    query += ` ORDER BY wp.created_at DESC LIMIT $${queryParams.length + 1} OFFSET $${queryP
    queryParams.push(limit, offset);

    const result = await client.query(query, queryParams);

    // Get total count
    const countResult = await client.query(
      `SELECT COUNT(*) FROM workflow_proposals WHERE tenant_id = $1 AND admin_status = ANY($
      [tenantId, statusFilter]
    );

    const response: GetProposalsResponse = {
      proposals: result.rows.map(row => ({
        ...row,
        sourcePattern: {
          id: row.source_pattern_id,
          patternName: row.pattern_name,
          detectedIntent: row.detected_intent,
        },
      })),
      total: parseInt(countResult.rows[0].count),
      hasMore: offset + result.rows.length < parseInt(countResult.rows[0].count),
    };

    return createResponse(200, response);

  } finally {
    client.release();
  }
};

// ============================================================================
```

```typescript
// GET /api/v2/admin/proposals/:id - Get single proposal
// ============================================================================

export const getProposal: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxyRes
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);
    const proposalId = event.pathParameters?.id;

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }

    if (!proposalId) {
      return createResponse(400, { error: 'Proposal ID required' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    // Get proposal
    const proposalResult = await client.query(
      `SELECT * FROM workflow_proposals WHERE id = $1 AND tenant_id = $2`,
      [proposalId, tenantId]
    );

    if (proposalResult.rows.length === 0) {
      return createResponse(404, { error: 'Proposal not found' });
    }

    // Get pattern
    const patternResult = await client.query(
      `SELECT * FROM neural_need_patterns WHERE id = $1`,
      [proposalResult.rows[0].source_pattern_id]
    );

    // Get evidence
    const evidenceResult = await client.query(
      `SELECT * FROM neural_need_evidence
       WHERE pattern_id = $1
       ORDER BY occurred_at DESC LIMIT 50`,
      [proposalResult.rows[0].source_pattern_id]
    );

    // Get review history
```

```
    const reviewsResult = await client.query(
      `SELECT wpr.*, u.email as reviewer_email
       FROM workflow_proposal_reviews wpr
       LEFT JOIN users u ON u.id = wpr.reviewer_id
       WHERE wpr.proposal_id = $1
       ORDER BY wpr.reviewed_at DESC`,
      [proposalId]
    );

    return createResponse(200, {
      proposal: proposalResult.rows[0],
      pattern: patternResult.rows[0],
      evidence: evidenceResult.rows,
      reviews: reviewsResult.rows,
    });

  } finally {
    client.release();
  }
};

// ============================================================================
// POST /api/v2/admin/proposals/:id/review - Review proposal
// ============================================================================

export const reviewProposal: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxy
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);
    const proposalId = event.pathParameters?.id;

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }

    if (!proposalId) {
      return createResponse(400, { error: 'Proposal ID required' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    const request: ReviewProposalRequest = JSON.parse(event.body || '{}');

    if (!request.action) {
```

```
  return createResponse(400, { error: 'Action required' });
}

// Fetch proposal
const proposalResult = await client.query<WorkflowProposal>(
  `SELECT * FROM workflow_proposals WHERE id = $1 AND tenant_id = $2`,
  [proposalId, tenantId]
);

if (proposalResult.rows.length === 0) {
  return createResponse(404, { error: 'Proposal not found' });
}

const proposal = proposalResult.rows[0];
const previousStatus = proposal.admin_status;

// Determine new status
let newStatus: string;
switch (request.action) {
  case 'approve':
    newStatus = 'approved';
    break;
  case 'decline':
    newStatus = 'declined';
    break;
  case 'request_test':
    newStatus = 'testing';
    break;
  case 'modify':
    newStatus = previousStatus; // Stay in current status
    break;
  default:
    return createResponse(400, { error: `Invalid action: ${request.action}` });
}

// Update proposal
await client.query(
  `UPDATE workflow_proposals SET
    admin_status = $2,
    admin_reviewer_id = $3,
    admin_review_timestamp = NOW(),
    admin_notes = COALESCE($4, admin_notes),
    admin_modifications = COALESCE($5, admin_modifications),
    updated_at = NOW()
  WHERE id = $1`,
  [
```

```
    proposalId,
    newStatus,
    userId,
    request.notes,
    request.modifications ? JSON.stringify(request.modifications) : null,
  ]
);

// Record review
const reviewResult = await client.query<{ id: string }>(
  `INSERT INTO workflow_proposal_reviews (
    proposal_id, reviewer_type, reviewer_id, action,
    previous_status, new_status, review_notes, modifications_made
  ) VALUES ($1, 'admin', $2, $3, $4, $5, $6, $7)
  RETURNING id`,
  [
    proposalId,
    userId,
    request.action,
    previousStatus,
    newStatus,
    request.notes,
    request.modifications ? JSON.stringify(request.modifications) : null,
  ]
);

// Audit log
await logAudit(client, {
  tenantId,
  userId,
  action: `proposal_${request.action}`,
  resourceType: 'workflow_proposal',
  resourceId: proposalId,
  details: {
    previousStatus,
    newStatus,
    notes: request.notes,
  },
});

const response: ReviewProposalResponse = {
  proposalId,
  newStatus: newStatus as any,
  reviewId: reviewResult.rows[0].id,
};
```

```
    return createResponse(200, response);

  } finally {
    client.release();
  }
};

// ============================================================================
// POST /api/v2/admin/proposals/:id/test - Test proposal
// ============================================================================

export const testProposal: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxyRe
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);
    const proposalId = event.pathParameters?.id;

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }

    if (!proposalId) {
      return createResponse(400, { error: 'Proposal ID required' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    const request: TestProposalRequest = JSON.parse(event.body || '{}');

    // Fetch proposal
    const proposalResult = await client.query<WorkflowProposal>(
      `SELECT * FROM workflow_proposals WHERE id = $1 AND tenant_id = $2`,
      [proposalId, tenantId]
    );

    if (proposalResult.rows.length === 0) {
      return createResponse(404, { error: 'Proposal not found' });
    }

    const proposal = proposalResult.rows[0];

    // Update status to testing
    await client.query(
      `UPDATE workflow_proposals SET
```

```
          admin_status = 'testing',
          test_status = 'testing',
          updated_at = NOW()
        WHERE id = $1`,
        [proposalId]
      );

      // Execute tests (simplified - in production, this would be async)
      const testResults = await executeProposalTests(
        proposal,
        request.testCases || [],
        request.iterations || 3
      );

      // Update with results
      const testStatus = testResults.testsPassed >= testResults.testsRun * 0.8 ? 'passed' : 'f

      await client.query(
        `UPDATE workflow_proposals SET
          test_status = $2,
          test_results = $3,
          admin_status = 'pending_admin',
          updated_at = NOW()
        WHERE id = $1`,
        [proposalId, testStatus, JSON.stringify(testResults)]
      );

      const response: TestProposalResponse = {
        proposalId,
        testStatus: testStatus as any,
        testResults,
        testExecutionIds: [],
      };

      return createResponse(200, response);

    } finally {
      client.release();
    }
  };

  async function executeProposalTests(
    proposal: WorkflowProposal,
    testCases: Array<{ input: string; expectedBehavior?: string }>,
    iterations: number
  ): Promise<WorkflowProposal['testResults']> {
```

```
  // Simplified test execution
  // In production, this would actually run the workflow in a sandbox

  const testsRun = Math.max(testCases.length, iterations);
  const testsPassed = Math.floor(testsRun * 0.85); // Simulated 85% pass rate

  return {
    testsRun,
    testsPassed,
    testsFailed: testsRun - testsPassed,
    avgLatencyMs: proposal.proposedWorkflow.metadata.estimatedLatencyMs * 1.1,
    avgQualityScore: proposal.neuralConfidence * 0.95,
    issues: testsRun > testsPassed ? ['Some edge cases produced lower quality results'] : []
  };
}

// ============================================================================
// POST /api/v2/admin/proposals/:id/publish - Publish approved proposal
// ============================================================================

export const publishProposal: APIGatewayProxyHandler = async (event): Promise<APIGatewayProx
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);
    const proposalId = event.pathParameters?.id;

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }

    if (!proposalId) {
      return createResponse(400, { error: 'Proposal ID required' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    const request: PublishProposalRequest = JSON.parse(event.body || '{}');

    // Fetch proposal
    const proposalResult = await client.query<WorkflowProposal>(
      `SELECT * FROM workflow_proposals WHERE id = $1 AND tenant_id = $2`,
      [proposalId, tenantId]
    );
```

```
if (proposalResult.rows.length === 0) {
  return createResponse(404, { error: 'Proposal not found' });
}

const proposal = proposalResult.rows[0];

if (proposal.admin_status !== 'approved') {
  return createResponse(400, { error: 'Proposal must be approved before publishing' });
}

// Create production workflow from proposal
const workflowResult = await client.query<{ id: string }>(
  `INSERT INTO production_workflows (
    tenant_id, name, description, category, workflow_type,
    dag_definition, complexity, created_by, source_proposal_id
  ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)
  RETURNING id`,
  [
    tenantId,
    proposal.proposal_name,
    proposal.proposal_description,
    request.publishToCategory || proposal.workflow_category,
    proposal.workflow_type,
    JSON.stringify(proposal.proposed_workflow),
    proposal.estimated_complexity,
    userId,
    proposalId,
  ]
);

const publishedWorkflowId = workflowResult.rows[0].id;

// Update proposal
await client.query(
  `UPDATE workflow_proposals SET
    published_workflow_id = $2,
    published_at = NOW(),
    updated_at = NOW()
  WHERE id = $1`,
  [proposalId, publishedWorkflowId]
);

// Update source pattern
await client.query(
  `UPDATE neural_need_patterns SET
    status = 'resolved',
```

```
        updated_at = NOW()
      WHERE id = $1`,
      [proposal.source_pattern_id]
    );

    // Audit log
    await logAudit(client, {
      tenantId,
      userId,
      action: 'proposal_published',
      resourceType: 'workflow_proposal',
      resourceId: proposalId,
      details: {
        publishedWorkflowId,
        workflowName: proposal.proposal_name,
      },
    });

    const response: PublishProposalResponse = {
      proposalId,
      publishedWorkflowId,
      publishedAt: new Date().toISOString(),
    };

    return createResponse(200, response);

  } finally {
    client.release();
  }
};

// ============================================================================
// GET/PUT /api/v2/admin/proposals/config - Threshold configuration
// ============================================================================

export const getThresholdConfig: APIGatewayProxyHandler = async (event): Promise<APIGatewayP
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }
```

```
    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    const configResult = await client.query(
      `SELECT * FROM proposal_threshold_config WHERE tenant_id = $1`,
      [tenantId]
    );

    const weightsResult = await client.query(
      `SELECT * FROM evidence_weight_config WHERE tenant_id = $1 ORDER BY evidence_type`,
      [tenantId]
    );

    return createResponse(200, {
      thresholds: configResult.rows[0] || {},
      evidenceWeights: weightsResult.rows,
    });

  } finally {
    client.release();
  }
};

export const updateThresholdConfig: APIGatewayProxyHandler = async (event): Promise<APIGatew
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);

    const request: UpdateThresholdConfigRequest = JSON.parse(event.body || '{}');

    // Build dynamic update query
    const updates: string[] = [];
    const values: any[] = [tenantId];
    let paramIndex = 2;

    for (const [key, value] of Object.entries(request.config)) {
      const snakeKey = key.replace(/([A-Z])/g, '_$1').toLowerCase();
      updates.push(`${snakeKey} = $${paramIndex}`);
      values.push(value);
```

```
      paramIndex++;
    }

    if (updates.length === 0) {
      return createResponse(400, { error: 'No configuration values provided' });
    }

    updates.push(`updated_at = NOW()`);
    updates.push(`updated_by = $${paramIndex}`);
    values.push(userId);

    await client.query(
      `INSERT INTO proposal_threshold_config (tenant_id) VALUES ($1)
       ON CONFLICT (tenant_id) DO UPDATE SET ${updates.join(', ')}`,
      values
    );

    // Audit log
    await logAudit(client, {
      tenantId,
      userId,
      action: 'threshold_config_updated',
      resourceType: 'proposal_config',
      resourceId: tenantId,
      details: request.config,
    });

    return createResponse(200, { success: true });

  } finally {
    client.release();
  }
};

export const updateEvidenceWeights: APIGatewayProxyHandler = async (event): Promise<APIGatew
  const client = await pool.connect();

  try {
    const tenantId = getTenantId(event);
    const userId = getUserId(event);

    if (!tenantId || !userId || !await isAdmin(client, tenantId, userId)) {
      return createResponse(403, { error: 'Admin access required' });
    }

    await client.query(`SET app.current_tenant_id = '${tenantId}'`);
```

```
    const request: UpdateEvidenceWeightsRequest = JSON.parse(event.body || '{}');

    for (const weight of request.weights) {
      await client.query(
        `INSERT INTO evidence_weight_config (tenant_id, evidence_type, weight, enabled, upda
         VALUES ($1, $2, $3, COALESCE($4, TRUE), $5)
         ON CONFLICT (tenant_id, evidence_type) DO UPDATE SET
           weight = $3,
           enabled = COALESCE($4, evidence_weight_config.enabled),
           updated_by = $5,
           updated_at = NOW()`,
        [tenantId, weight.evidenceType, weight.weight, weight.enabled, userId]
      );
    }

    // Audit log
    await logAudit(client, {
      tenantId,
      userId,
      action: 'evidence_weights_updated',
      resourceType: 'evidence_config',
      resourceId: tenantId,
      details: { weights: request.weights },
    });

    return createResponse(200, { success: true });

  } finally {
    client.release();
  }
};
```

39.10 React Admin Dashboard Components

```
// ============================================================================
// packages/admin-dashboard/src/components/proposals/ProposalQueuePage.tsx
// Main proposal review queue page
// ============================================================================

import React, { useState, useEffect } from 'react';
import {
  Box,
  Card,
```

```
  CardContent,
  Typography,
  Chip,
  Button,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  IconButton,
  Tooltip,
  LinearProgress,
  Badge,
  Tab,
  Tabs,
} from '@mui/material';
import {
  CheckCircle,
  Cancel,
  PlayArrow,
  Visibility,
  Warning,
  TrendingUp,
  People,
  Timer,
} from '@mui/icons-material';
import { useNavigate } from 'react-router-dom';
import { useProposals, useProposalStats } from '../../hooks/useProposals';
import { WorkflowProposal, ProposalAdminStatus } from '@radiant/core/types/workflow-proposal

const statusColors: Record<ProposalAdminStatus, 'default' | 'primary' | 'secondary' | 'erro
  pending_brain: 'default',
  pending_admin: 'warning',
  approved: 'success',
  declined: 'error',
  testing: 'info',
};

const statusLabels: Record<ProposalAdminStatus, string> = {
  pending_brain: 'Brain Review',
  pending_admin: 'Needs Review',
  approved: 'Approved',
  declined: 'Declined',
  testing: 'Testing',
};
```

```
export const ProposalQueuePage: React.FC = () => {
  const navigate = useNavigate();
  const [statusFilter, setStatusFilter] = useState<ProposalAdminStatus>('pending_admin');
  const { proposals, loading, refetch } = useProposals({ status: [statusFilter] });
  const { stats } = useProposalStats();

  return (
    <Box sx={{ p: 3 }}>
      {/* Header Stats */}
      <Box sx={{ display: 'flex', gap: 2, mb: 3 }}>
        <StatCard
          title="Pending Review"
          value={stats?.pendingAdmin || 0}
          icon={<Warning color="warning" />}
          color="warning"
        />
        <StatCard
          title="Approved Today"
          value={stats?.approvedToday || 0}
          icon={<CheckCircle color="success" />}
          color="success"
        />
        <StatCard
          title="Users Affected"
          value={stats?.totalUsersAffected || 0}
          icon={<People color="primary" />}
          color="primary"
        />
        <StatCard
          title="Avg Confidence"
          value={`${((stats?.avgConfidence || 0) * 100).toFixed(0)}%`}
          icon={<TrendingUp color="info" />}
          color="info"
        />
      </Box>

      {/* Status Tabs */}
      <Card sx={{ mb: 3 }}>
        <Tabs
          value={statusFilter}
          onChange={(_, v) => setStatusFilter(v)}
          indicatorColor="primary"
        >
          <Tab
            label={
```

```
        <Badge badgeContent={stats?.pendingAdmin || 0} color="warning">
          Pending Review
        </Badge>
      }
      value="pending_admin"
    />
    <Tab label="Testing" value="testing" />
    <Tab label="Approved" value="approved" />
    <Tab label="Declined" value="declined" />
  </Tabs>
</Card>

{/* Proposals Table */}
<Card>
  <CardContent>
    {loading ? (
      <LinearProgress />
    ) : (
      <TableContainer>
        <Table>
          <TableHead>
            <TableRow>
              <TableCell>Proposal</TableCell>
              <TableCell>Category</TableCell>
              <TableCell align="center">Evidence</TableCell>
              <TableCell align="center">Users</TableCell>
              <TableCell align="center">Confidence</TableCell>
              <TableCell align="center">Risk</TableCell>
              <TableCell align="center">Status</TableCell>
              <TableCell align="right">Actions</TableCell>
            </TableRow>
          </TableHead>
          <TableBody>
            {proposals.map((proposal) => (
              <ProposalRow
                key={proposal.id}
                proposal={proposal}
                onView={() => navigate(`/proposals/${proposal.id}`)}
                onRefresh={refetch}
              />
            ))}
            {proposals.length === 0 && (
              <TableRow>
                <TableCell colSpan={8} align="center">
                  <Typography color="textSecondary">
                    No proposals in this status
```

```
                      </Typography>
                    </TableCell>
                  </TableRow>
                )}
              </TableBody>
            </Table>
          </TableContainer>
        )}
      </CardContent>
    </Card>
  </Box>
 );
};

interface ProposalRowProps {
  proposal: WorkflowProposal;
  onView: () => void;
  onRefresh: () => void;
}

const ProposalRow: React.FC<ProposalRowProps> = ({ proposal, onView, onRefresh }) => {
  const riskLevel = proposal.brainRiskAssessment?.overallRisk || 0;

  return (
    <TableRow hover>
      <TableCell>
        <Typography variant="subtitle2">{proposal.proposalCode}</Typography>
        <Typography variant="body2" color="textSecondary" noWrap sx={{ maxWidth: 300 }}>
          {proposal.proposalName}
        </Typography>
      </TableCell>
      <TableCell>
        <Chip
          label={proposal.workflowCategory || 'General'}
          size="small"
          variant="outlined"
        />
      </TableCell>
      <TableCell align="center">
        <Typography variant="body2">{proposal.evidenceCount}</Typography>
      </TableCell>
      <TableCell align="center">
        <Typography variant="body2">{proposal.uniqueUsersAffected}</Typography>
      </TableCell>
      <TableCell align="center">
        <ConfidenceChip value={proposal.neuralConfidence} />
```

```
      </TableCell>
      <TableCell align="center">
        <RiskChip value={riskLevel} />
      </TableCell>
      <TableCell align="center">
        <Chip
          label={statusLabels[proposal.adminStatus]}
          color={statusColors[proposal.adminStatus]}
          size="small"
        />
      </TableCell>
      <TableCell align="right">
        <Tooltip title="View Details">
          <IconButton size="small" onClick={onView}>
            <Visibility />
          </IconButton>
        </Tooltip>
      </TableCell>
    </TableRow>
  );
};

const StatCard: React.FC<{
  title: string;
  value: number | string;
  icon: React.ReactNode;
  color: string;
}> = ({ title, value, icon, color }) => (
  <Card sx={{ flex: 1 }}>
    <CardContent sx={{ display: 'flex', alignItems: 'center', gap: 2 }}>
      {icon}
      <Box>
        <Typography variant="h4">{value}</Typography>
        <Typography variant="body2" color="textSecondary">{title}</Typography>
      </Box>
    </CardContent>
  </Card>
);

const ConfidenceChip: React.FC<{ value: number }> = ({ value }) => {
  const percent = Math.round(value * 100);
  const color = percent >= 80 ? 'success' : percent >= 60 ? 'warning' : 'error';

  return (
    <Chip
      label={`${percent}%`}
```

```
      color={color}
      size="small"
      variant="outlined"
    />
  );
};

const RiskChip: React.FC<{ value: number }> = ({ value }) => {
  const percent = Math.round(value * 100);
  const color = percent <= 20 ? 'success' : percent <= 40 ? 'warning' : 'error';
  const label = percent <= 20 ? 'Low' : percent <= 40 ? 'Medium' : 'High';

  return (
    <Chip
      label={label}
      color={color}
      size="small"
      variant="outlined"
    />
  );
};

export default ProposalQueuePage;
```

```
// ============================================================================
// packages/admin-dashboard/src/components/proposals/ProposalDetailPage.tsx
// Detailed proposal review page with workflow visualization
// ============================================================================

import React, { useState } from 'react';
import {
  Box,
  Card,
  CardContent,
  Typography,
  Button,
  Grid,
  Divider,
  TextField,
  Dialog,
  DialogTitle,
  DialogContent,
  DialogActions,
  Stepper,
  Step,
  StepLabel,
```

```
  Alert,
  Chip,
  List,
  ListItem,
  ListItemText,
  ListItemIcon,
  Paper,
} from '@mui/material';
import {
  CheckCircle,
  Cancel,
  PlayArrow,
  Publish,
  Warning,
  Info,
  TrendingUp,
  People,
  AccessTime,
  Category,
} from '@mui/icons-material';
import { useParams, useNavigate } from 'react-router-dom';
import { useProposalDetail, useReviewProposal, useTestProposal, usePublishProposal } from '
import { WorkflowDAGViewer } from '../workflows/WorkflowDAGViewer';
import { EvidenceTimeline } from './EvidenceTimeline';

export const ProposalDetailPage: React.FC = () => {
  const { id } = useParams<{ id: string }>();
  const navigate = useNavigate();
  const { proposal, pattern, evidence, reviews, loading, refetch } = useProposalDetail(id!);
  const { review, reviewing } = useReviewProposal();
  const { test, testing } = useTestProposal();
  const { publish, publishing } = usePublishProposal();

  const [reviewDialogOpen, setReviewDialogOpen] = useState(false);
  const [reviewAction, setReviewAction] = useState<'approve' | 'decline'>('approve');
  const [reviewNotes, setReviewNotes] = useState('');

  if (loading || !proposal) {
    return <Box sx={{ p: 3 }}>Loading...</Box>;
  }

  const handleReview = async () => {
    await review(id!, { action: reviewAction, notes: reviewNotes });
    setReviewDialogOpen(false);
    refetch();
  };
```

```
const handleTest = async () => {
  await test(id!, { testMode: 'automated', iterations: 5 });
  refetch();
};

const handlePublish = async () => {
  await publish(id!, {});
  navigate('/proposals');
};

return (
  <Box sx={{ p: 3 }}>
    {/* Header */}
    <Box sx={{ display: 'flex', justifyContent: 'space-between', mb: 3 }}>
      <Box>
        <Typography variant="h4">{proposal.proposalCode}</Typography>
        <Typography variant="h6" color="textSecondary">{proposal.proposalName}</Typography
      </Box>
      <Box sx={{ display: 'flex', gap: 1 }}>
        {proposal.adminStatus === 'pending_admin' && (
          <>
            <Button
              variant="contained"
              color="success"
              startIcon={<CheckCircle />}
              onClick={() => { setReviewAction('approve'); setReviewDialogOpen(true); }}
            >
              Approve
            </Button>
            <Button
              variant="outlined"
              color="error"
              startIcon={<Cancel />}
              onClick={() => { setReviewAction('decline'); setReviewDialogOpen(true); }}
            >
              Decline
            </Button>
            <Button
              variant="outlined"
              startIcon={<PlayArrow />}
              onClick={handleTest}
              disabled={testing}
            >
              Run Tests
            </Button>
```

```
      </>
    )}
    {proposal.adminStatus === 'approved' && !proposal.publishedWorkflowId && (
      <Button
        variant="contained"
        color="primary"
        startIcon={<Publish />}
        onClick={handlePublish}
        disabled={publishing}
      >
        Publish to Production
      </Button>
    )}
  </Box>
</Box>

<Grid container spacing={3}>
  {/* Left Column - Details */}
  <Grid item xs={12} md={4}>
    {/* Summary Card */}
    <Card sx={{ mb: 2 }}>
      <CardContent>
        <Typography variant="h6" gutterBottom>Summary</Typography>
        <List dense>
          <ListItem>
            <ListItemIcon><Category /></ListItemIcon>
            <ListItemText
              primary="Category"
              secondary={proposal.workflowCategory || 'General'}
            />
          </ListItem>
          <ListItem>
            <ListItemIcon><People /></ListItemIcon>
            <ListItemText
              primary="Users Affected"
              secondary={proposal.uniqueUsersAffected}
            />
          </ListItem>
          <ListItem>
            <ListItemIcon><TrendingUp /></ListItemIcon>
            <ListItemText
              primary="Evidence Count"
              secondary={proposal.evidenceCount}
            />
          </ListItem>
          <ListItem>
```

```
        <ListItemIcon><AccessTime /></ListItemIcon>
        <ListItemText
          primary="Time Span"
          secondary={`${proposal.evidenceTimeSpanDays} days`}
        />
      </ListItem>
    </List>
  </CardContent>
</Card>

{/* Risk Assessment Card */}
{proposal.brainRiskAssessment && (
  <Card sx={{ mb: 2 }}>
    <CardContent>
      <Typography variant="h6" gutterBottom>Risk Assessment</Typography>
      <RiskBar label="Cost Risk" value={proposal.brainRiskAssessment.costRisk} />
      <RiskBar label="Latency Risk" value={proposal.brainRiskAssessment.latencyRis
      <RiskBar label="Quality Risk" value={proposal.brainRiskAssessment.qualityRis
      <RiskBar label="Compliance Risk" value={proposal.brainRiskAssessment.complia
      <Divider sx={{ my: 1 }} />
      <RiskBar label="Overall Risk" value={proposal.brainRiskAssessment.overallRis

      {proposal.brainRiskAssessment.riskFactors.length > 0 && (
        <Box sx={{ mt: 2 }}>
          <Typography variant="subtitle2" color="warning.main">Risk Factors:</Typo
          {proposal.brainRiskAssessment.riskFactors.map((factor, i) => (
            <Chip key={i} label={factor} size="small" sx={{ mr: 0.5, mt: 0.5 }} />
          ))}
        </Box>
      )}
    </CardContent>
  </Card>
)}

{/* Neural Reasoning */}
<Card sx={{ mb: 2 }}>
  <CardContent>
    <Typography variant="h6" gutterBottom>Neural Engine Reasoning</Typography>
    <Typography variant="body2" sx={{ whiteSpace: 'pre-line' }}>
      {proposal.neuralReasoning}
    </Typography>
  </CardContent>
</Card>

{/* Test Results */}
{proposal.testResults && (
```

```
    <Card sx={{ mb: 2 }}>
      <CardContent>
        <Typography variant="h6" gutterBottom>Test Results</Typography>
        <Alert
          severity={proposal.testStatus === 'passed' ? 'success' : 'warning'}
          sx={{ mb: 2 }}
        >
          {proposal.testResults.testsPassed} / {proposal.testResults.testsRun} tests
        </Alert>
        <Typography variant="body2">
          Avg Latency: {proposal.testResults.avgLatencyMs}ms
        </Typography>
        <Typography variant="body2">
          Avg Quality: {(proposal.testResults.avgQualityScore * 100).toFixed(0)}%
        </Typography>
        {proposal.testResults.issues.length > 0 && (
          <Box sx={{ mt: 1 }}>
            <Typography variant="subtitle2" color="warning.main">Issues:</Typography
            {proposal.testResults.issues.map((issue, i) => (
              <Typography key={i} variant="body2" color="textSecondary">• {issue}</T
            ))}
          </Box>
        )}
      </CardContent>
    </Card>
  )}
</Grid>

{/* Right Column - Workflow & Evidence */}
<Grid item xs={12} md={8}>
  {/* Workflow Visualization */}
  <Card sx={{ mb: 2 }}>
    <CardContent>
      <Typography variant="h6" gutterBottom>Proposed Workflow</Typography>
      <Box sx={{ height: 400, border: '1px solid #e0e0e0', borderRadius: 1 }}>
        <WorkflowDAGViewer dag={proposal.proposedWorkflow} />
      </Box>
      <Box sx={{ mt: 2, display: 'flex', gap: 2 }}>
        <Chip
          icon={<AccessTime />}
          label={`Est. Latency: ${proposal.proposedWorkflow.metadata.estimatedLatenc
          variant="outlined"
        />
        <Chip
          icon={<TrendingUp />}
          label={`Est. Cost: $${proposal.proposedWorkflow.metadata.estimatedCostPer1
```

```
            variant="outlined"
          />
          <Chip
            label={`${proposal.proposedWorkflow.nodes.length} nodes`}
            variant="outlined"
          />
        </Box>
      </CardContent>
    </Card>

    {/* Evidence Timeline */}
    <Card>
      <CardContent>
        <Typography variant="h6" gutterBottom>Evidence Timeline</Typography>
        <EvidenceTimeline evidence={evidence} />
      </CardContent>
    </Card>
  </Grid>
</Grid>

{/* Review Dialog */}
<Dialog open={reviewDialogOpen} onClose={() => setReviewDialogOpen(false)} maxWidth="s
  <DialogTitle>
    {reviewAction === 'approve' ? 'Approve Proposal' : 'Decline Proposal'}
  </DialogTitle>
  <DialogContent>
    <TextField
      label="Review Notes"
      multiline
      rows={4}
      fullWidth
      value={reviewNotes}
      onChange={(e) => setReviewNotes(e.target.value)}
      placeholder={reviewAction === 'approve'
        ? 'Optional: Add any notes about this approval...'
        : 'Please explain why this proposal is being declined...'}
      sx={{ mt: 2 }}
    />
  </DialogContent>
  <DialogActions>
    <Button onClick={() => setReviewDialogOpen(false)}>Cancel</Button>
    <Button
      onClick={handleReview}
      color={reviewAction === 'approve' ? 'success' : 'error'}
      variant="contained"
      disabled={reviewing}
```

```
          >
            {reviewAction === 'approve' ? 'Approve' : 'Decline'}
          </Button>
        </DialogActions>
      </Dialog>
    </Box>
  );
};

const RiskBar: React.FC<{ label: string; value: number; bold?: boolean }> = ({ label, value,
  const percent = Math.round(value * 100);
  const color = percent <= 20 ? '#4caf50' : percent <= 40 ? '#ff9800' : '#f44336';

  return (
    <Box sx={{ mb: 1 }}>
      <Box sx={{ display: 'flex', justifyContent: 'space-between' }}>
        <Typography variant={bold ? 'subtitle2' : 'body2'}>{label}</Typography>
        <Typography variant={bold ? 'subtitle2' : 'body2'}>{percent}%</Typography>
      </Box>
      <Box sx={{ width: '100%', height: 8, bgcolor: '#e0e0e0', borderRadius: 1 }}>
        <Box sx={{ width: `${percent}%`, height: '100%', bgcolor: color, borderRadius: 1 }}
      </Box>
    </Box>
  );
};

export default ProposalDetailPage;
```

```
// ============================================================================
// packages/admin-dashboard/src/components/proposals/ProposalConfigPage.tsx
// Admin configuration page for thresholds and evidence weights
// ============================================================================

import React, { useState, useEffect } from 'react';
import {
  Box,
  Card,
  CardContent,
  Typography,
  Grid,
  TextField,
  Slider,
  Switch,
  FormControlLabel,
  Button,
  Divider,
```

```
  Alert,
  Tooltip,
  IconButton,
} from '@mui/material';
import { Save, Refresh, Info } from '@mui/icons-material';
import { useThresholdConfig, useUpdateThresholdConfig, useUpdateEvidenceWeights } from '../

export const ProposalConfigPage: React.FC = () => {
  const { config, evidenceWeights, loading, refetch } = useThresholdConfig();
  const { update: updateThresholds, updating: updatingThresholds } = useUpdateThresholdConfi
  const { update: updateWeights, updating: updatingWeights } = useUpdateEvidenceWeights();

  const [thresholdForm, setThresholdForm] = useState(config || {});
  const [weightsForm, setWeightsForm] = useState(evidenceWeights || []);
  const [hasChanges, setHasChanges] = useState(false);

  useEffect(() => {
    if (config) setThresholdForm(config);
    if (evidenceWeights) setWeightsForm(evidenceWeights);
  }, [config, evidenceWeights]);

  const handleThresholdChange = (key: string, value: any) => {
    setThresholdForm(prev => ({ ...prev, [key]: value }));
    setHasChanges(true);
  };

  const handleWeightChange = (evidenceType: string, value: number) => {
    setWeightsForm(prev =>
      prev.map(w => w.evidenceType === evidenceType ? { ...w, weight: value } : w)
    );
    setHasChanges(true);
  };

  const handleSave = async () => {
    await updateThresholds({ config: thresholdForm });
    await updateWeights({ weights: weightsForm });
    setHasChanges(false);
    refetch();
  };

  if (loading) return <Box sx={{ p: 3 }}>Loading...</Box>;

  return (
    <Box sx={{ p: 3 }}>
      <Box sx={{ display: 'flex', justifyContent: 'space-between', mb: 3 }}>
        <Typography variant="h4">Proposal System Configuration</Typography>
```

```
  <Box>
    <Button startIcon={<Refresh />} onClick={refetch} sx={{ mr: 1 }}>
      Reset
    </Button>
    <Button
      variant="contained"
      startIcon={<Save />}
      onClick={handleSave}
      disabled={!hasChanges || updatingThresholds || updatingWeights}
    >
      Save Changes
    </Button>
  </Box>
</Box>

{hasChanges && (
  <Alert severity="info" sx={{ mb: 3 }}>
    You have unsaved changes. Click "Save Changes" to apply them.
  </Alert>
)}

<Grid container spacing={3}>
  {/* Occurrence Thresholds */}
  <Grid item xs={12} md={6}>
    <Card>
      <CardContent>
        <Typography variant="h6" gutterBottom>Occurrence Thresholds</Typography>
        <Typography variant="body2" color="textSecondary" sx={{ mb: 2 }}>
          Minimum requirements for a need pattern to qualify for proposal generation
        </Typography>

        <ConfigSlider
          label="Minimum Evidence Count"
          value={thresholdForm.minEvidenceCount || 5}
          onChange={(v) => handleThresholdChange('minEvidenceCount', v)}
          min={1}
          max={50}
          tooltip="Number of evidence records needed before proposing a workflow"
        />

        <ConfigSlider
          label="Minimum Unique Users"
          value={thresholdForm.minUniqueUsers || 3}
          onChange={(v) => handleThresholdChange('minUniqueUsers', v)}
          min={1}
          max={20}
```

```
          tooltip="Number of different users who must encounter the issue"
        />

        <ConfigSlider
          label="Minimum Time Span (hours)"
          value={thresholdForm.minTimeSpanHours || 24}
          onChange={(v) => handleThresholdChange('minTimeSpanHours', v)}
          min={1}
          max={168}
          tooltip="Evidence must span at least this many hours"
        />
      </CardContent>
    </Card>
  </Grid>

  {/* Impact Thresholds */}
  <Grid item xs={12} md={6}>
    <Card>
      <CardContent>
        <Typography variant="h6" gutterBottom>Impact Thresholds</Typography>
        <Typography variant="body2" color="textSecondary" sx={{ mb: 2 }}>
          Score requirements measuring severity of the identified need
        </Typography>

        <ConfigSlider
          label="Minimum Total Evidence Score"
          value={(thresholdForm.minTotalEvidenceScore || 0.6) * 100}
          onChange={(v) => handleThresholdChange('minTotalEvidenceScore', v / 100)}
          min={10}
          max={100}
          format={(v) => `${v}%`}
          tooltip="Cumulative weighted score from all evidence"
        />

        <ConfigSlider
          label="Minimum Neural Confidence"
          value={(thresholdForm.minNeuralConfidence || 0.75) * 100}
          onChange={(v) => handleThresholdChange('minNeuralConfidence', v / 100)}
          min={50}
          max={100}
          format={(v) => `${v}%`}
          tooltip="Neural Engine's minimum confidence to generate a proposal"
        />
      </CardContent>
    </Card>
  </Grid>
```

```
{/* Brain Governor Thresholds */}
<Grid item xs={12} md={6}>
  <Card>
    <CardContent>
      <Typography variant="h6" gutterBottom>Brain Governor Thresholds</Typography>
      <Typography variant="body2" color="textSecondary" sx={{ mb: 2 }}>
        Maximum acceptable risk levels for Brain to approve proposals
      </Typography>

      <ConfigSlider
        label="Max Cost Risk"
        value={(thresholdForm.maxCostRisk || 0.3) * 100}
        onChange={(v) => handleThresholdChange('maxCostRisk', v / 100)}
        min={0}
        max={100}
        format={(v) => `${v}%`}
        tooltip="Proposals exceeding this cost risk will be vetoed"
      />

      <ConfigSlider
        label="Max Latency Risk"
        value={(thresholdForm.maxLatencyRisk || 0.4) * 100}
        onChange={(v) => handleThresholdChange('maxLatencyRisk', v / 100)}
        min={0}
        max={100}
        format={(v) => `${v}%`}
        tooltip="Proposals exceeding this latency risk will be vetoed"
      />

      <ConfigSlider
        label="Max Compliance Risk"
        value={(thresholdForm.maxComplianceRisk || 0.1) * 100}
        onChange={(v) => handleThresholdChange('maxComplianceRisk', v / 100)}
        min={0}
        max={50}
        format={(v) => `${v}%`}
        tooltip="Proposals exceeding this compliance risk will be vetoed"
      />
    </CardContent>
  </Card>
</Grid>

{/* Rate Limiting */}
<Grid item xs={12} md={6}>
  <Card>
```

```
    <CardContent>
      <Typography variant="h6" gutterBottom>Rate Limiting</Typography>
      <Typography variant="body2" color="textSecondary" sx={{ mb: 2 }}>
        Limits on proposal generation frequency
      </Typography>

      <ConfigSlider
        label="Max Proposals Per Day"
        value={thresholdForm.maxProposalsPerDay || 10}
        onChange={(v) => handleThresholdChange('maxProposalsPerDay', v)}
        min={1}
        max={50}
        tooltip="Maximum new proposals that can be generated daily"
      />

      <ConfigSlider
        label="Max Proposals Per Week"
        value={thresholdForm.maxProposalsPerWeek || 30}
        onChange={(v) => handleThresholdChange('maxProposalsPerWeek', v)}
        min={1}
        max={200}
        tooltip="Maximum new proposals that can be generated weekly"
      />

      <ConfigSlider
        label="Cooldown After Decline (hours)"
        value={thresholdForm.cooldownAfterDeclineHours || 72}
        onChange={(v) => handleThresholdChange('cooldownAfterDeclineHours', v)}
        min={1}
        max={168}
        tooltip="Wait time before regenerating proposal for a declined pattern"
      />
    </CardContent>
  </Card>
</Grid>

{/* Evidence Weights */}
<Grid item xs={12}>
  <Card>
    <CardContent>
      <Typography variant="h6" gutterBottom>Evidence Weights</Typography>
      <Typography variant="body2" color="textSecondary" sx={{ mb: 2 }}>
        How much each type of evidence contributes to the total score
      </Typography>

      <Grid container spacing={2}>
```

```
                {weightsForm.map((weight) => (
                  <Grid item xs={12} sm={6} md={4} key={weight.evidenceType}>
                    <ConfigSlider
                      label={formatEvidenceType(weight.evidenceType)}
                      value={weight.weight * 100}
                      onChange={(v) => handleWeightChange(weight.evidenceType, v / 100)}
                      min={0}
                      max={100}
                      format={(v) => `${v}%`}
                      tooltip={weight.description}
                    />
                  </Grid>
                ))}
              </Grid>
            </CardContent>
          </Card>
        </Grid>
      </Grid>
    </Box>
  );
};

interface ConfigSliderProps {
  label: string;
  value: number;
  onChange: (value: number) => void;
  min: number;
  max: number;
  format?: (value: number) => string;
  tooltip?: string;
}

const ConfigSlider: React.FC<ConfigSliderProps> = ({
  label,
  value,
  onChange,
  min,
  max,
  format = (v) => String(v),
  tooltip,
}) => (
  <Box sx={{ mb: 3 }}>
    <Box sx={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center' }}>
      <Box sx={{ display: 'flex', alignItems: 'center' }}>
        <Typography variant="subtitle2">{label}</Typography>
        {tooltip && (
```

```
          <Tooltip title={tooltip}>
            <IconButton size="small" sx={{ ml: 0.5 }}>
              <Info fontSize="small" />
            </IconButton>
          </Tooltip>
        )}
      </Box>
      <Typography variant="body2" color="primary">{format(value)}</Typography>
    </Box>
    <Slider
      value={value}
      onChange={(_, v) => onChange(v as number)}
      min={min}
      max={max}
      valueLabelDisplay="auto"
      valueLabelFormat={format}
    />
  </Box>
);

function formatEvidenceType(type: string): string {
  return type
    .split('_')
    .map(word => word.charAt(0).toUpperCase() + word.slice(1))
    .join(' ');
}

export default ProposalConfigPage;
```

SECTION-40-APP-ISOLATION

SECTION 40: APPLICATION-LEVEL DATA ISOLATION (v4.6.0)

> **CRITICAL: This section implements complete isolation between client apps (Think Tank, Launch Board, AlwaysMe, Mechanical Maker) AND Think Tank. Same user email in different apps = completely separate identities and data.**

40.1 ARCHITECTURE OVERVIEW

The Problem

Before v4.6.0, RADIANT had **tenant-level isolation** but not **application-level isolation**:

```
BEFORE (v4.5.0 and earlier):

 Tenant: Acme Corp (tenant_id: abc-123)


  User: alice@acme.com (single user record)
    Can access Think Tank data
    Can access Launch Board data     ← PROBLEM!
    Can access Think Tank data           ← PROBLEM!
    All apps share same chat history, preferences, etc.

  RLS Policy: WHERE tenant_id = current_setting('app.current_tenant_id')
    Only filters by tenant, not by app
```

The Solution

v4.6.0 implements **application-level isolation**:

```
AFTER (v4.6.0):

 Tenant: Acme Corp (tenant_id: abc-123)



   App: Think Tank (app_id: thinktank)

    AppUser: alice@acme.com (app_user_id: usr-tt-001)
      Chats, preferences, history ONLY for Think Tank




   App: Launch Board (app_id: launchboard)

    AppUser: alice@acme.com (app_user_id: usr-lb-002)
      Chats, preferences, history ONLY for Launch Board
```

```
  App: Think Tank (app_id: thinktank)

    AppUser: alice@acme.com (app_user_id: usr-sv-003)
       Chats, preferences, history ONLY for Think Tank
       COMPLETELY ISOLATED from Think Tank and Launch Board



RLS Policy: WHERE tenant_id = :tenant AND app_id = :app
   Filters by BOTH tenant AND app
```

Isolation Layers

```
                        RADIANT v4.6.0 ISOLATION LAYERS


LAYER 1: COGNITO (Identity)

  Think Tank Pool       Launch Board Pool     Think Tank Pool
  (thinktank-prod)      (launchboard-prod)    (thinktank-prod)

  custom:app_id =       custom:app_id =       custom:app_id =
  "thinktank"           "launchboard"         "thinktank"



LAYER 2: API GATEWAY (Routing)

  thinktank.domain.com    → thinktank API
  launchboard.domain.com  → launchboard API
  thinktank.domain.com  → thinktank API

  JWT Validation: Verify app_id in token matches route



LAYER 3: LAMBDA (Context)

  AuthContext = {
    tenantId: 'abc-123',
    appId: 'thinktank',     ← NEW: App ID extracted from JWT
    userId: 'usr-tt-001',   ← App-scoped user ID
```

```
    email: 'alice@acme.com'
  }

  Database Connection:
    SET app.current_tenant_id = 'abc-123';
    SET app.current_app_id = 'thinktank';   ← NEW


LAYER 4: DATABASE (RLS)

  CREATE POLICY app_isolation ON user_data
    USING (
      tenant_id = current_setting('app.current_tenant_id')::UUID
      AND
      app_id = current_setting('app.current_app_id')
    );

  Every SELECT/INSERT/UPDATE/DELETE filtered by BOTH
```

40.2 DATABASE SCHEMA CHANGES

Migration: 040_app_level_isolation.sql

```sql
-- ============================================================================
-- RADIANT v4.6.0 - Application-Level Data Isolation
-- ============================================================================
-- This migration adds app_id isolation to all user-facing tables
-- ============================================================================

-- Create function to get current app_id from session
CREATE OR REPLACE FUNCTION current_app_id() RETURNS VARCHAR(50) AS $$
BEGIN
    RETURN NULLIF(current_setting('app.current_app_id', true), '');
EXCEPTION WHEN OTHERS THEN RETURN NULL;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER STABLE;

-- ============================================================================
-- APP_USERS: App-Scoped User Instances
-- ============================================================================
-- A single email can have MULTIPLE app_user records (one per app)
```

```sql
-- This is the PRIMARY user identity within each application

CREATE TABLE IF NOT EXISTS app_users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Multi-tenant + Multi-app isolation
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    app_id VARCHAR(50) NOT NULL,

    -- Link to base user (optional - for cross-app identity correlation by admins only)
    base_user_id UUID REFERENCES users(id) ON DELETE SET NULL,

    -- Identity (same email can exist in multiple apps)
    cognito_sub VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL,
    email_verified BOOLEAN DEFAULT false,

    -- Profile (app-specific)
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    display_name VARCHAR(200),
    avatar_url TEXT,

    -- App-specific preferences
    preferences JSONB DEFAULT '{}',
    timezone VARCHAR(50) DEFAULT 'UTC',
    locale VARCHAR(10) DEFAULT 'en-US',

    -- Status
    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'su
    last_login_at TIMESTAMPTZ,
    login_count INTEGER DEFAULT 0,

    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    deleted_at TIMESTAMPTZ,

    -- Unique constraint: one user per email per app per tenant
    UNIQUE(tenant_id, app_id, email),
    -- Cognito sub is unique per app
    UNIQUE(app_id, cognito_sub)
);

-- Indexes for app_users
CREATE INDEX idx_app_users_tenant_app ON app_users(tenant_id, app_id);
```

```sql
CREATE INDEX idx_app_users_email ON app_users(email);
CREATE INDEX idx_app_users_cognito ON app_users(cognito_sub);
CREATE INDEX idx_app_users_base_user ON app_users(base_user_id);
CREATE INDEX idx_app_users_status ON app_users(status) WHERE deleted_at IS NULL;

-- RLS for app_users
ALTER TABLE app_users ENABLE ROW LEVEL SECURITY;

CREATE POLICY app_users_isolation ON app_users
    FOR ALL
    USING (
        tenant_id = current_tenant_id()
        AND app_id = current_app_id()
    );

-- Admin policy: Platform admins can view all users within their tenant
CREATE POLICY app_users_admin_view ON app_users
    FOR SELECT
    USING (
        tenant_id = current_tenant_id()
        AND current_setting('app.is_admin', true) = 'true'
    );

-- ============================================================================
-- ADD app_id TO EXISTING TABLES
-- ============================================================================

-- Add app_id column to user-facing tables that need isolation
-- Using IF NOT EXISTS pattern for idempotent migrations

-- Chats / Conversations (Think Tank and Client Apps)
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'thinktank_chats' AND column_name = 'app_id') THEN
        ALTER TABLE thinktank_chats ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktar
    END IF;
END $$;

DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'thinktank_messages' AND column_name = 'app_id') THEN
        ALTER TABLE thinktank_messages ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'think
    END IF;
END $$;
```

```
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'thinktank_conversations' AND column_name = 'app_id')
        ALTER TABLE thinktank_conversations ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT '
    END IF;
END $$;

-- Concurrent Sessions
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'concurrent_sessions' AND column_name = 'app_id') THEN
        ALTER TABLE concurrent_sessions ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thin
    END IF;
END $$;

-- Voice Sessions
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'voice_sessions' AND column_name = 'app_id') THEN
        ALTER TABLE voice_sessions ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank
    END IF;
END $$;

-- Memory
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'memory_stores' AND column_name = 'app_id') THEN
        ALTER TABLE memory_stores ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank'
    END IF;

    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'memories' AND column_name = 'app_id') THEN
        ALTER TABLE memories ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank';
    END IF;
END $$;

-- Canvases & Artifacts
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'canvases' AND column_name = 'app_id') THEN
```

```
        ALTER TABLE canvases ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'artifacts' AND column_name = 'app_id') THEN
        ALTER TABLE artifacts ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank';
    END IF;
END $$;

-- User Personas
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'user_personas' AND column_name = 'app_id') THEN
        ALTER TABLE user_personas ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank'
    END IF;
END $$;

-- Scheduled Prompts
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'scheduled_prompts' AND column_name = 'app_id') THEN
        ALTER TABLE scheduled_prompts ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinkt
    END IF;
END $$;

-- User Preferences (Neural Engine)
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'user_preferences' AND column_name = 'app_id') THEN
        ALTER TABLE user_preferences ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinkta
    END IF;
END $$;

-- User Memory (Neural Engine)
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'user_memory' AND column_name = 'app_id') THEN
        ALTER TABLE user_memory ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank';
    END IF;
END $$;

-- User Behavior Patterns
```

```
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'user_behavior_patterns' AND column_name = 'app_id') T
        ALTER TABLE user_behavior_patterns ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 't
    END IF;
END $$;

-- Feedback tables
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'feedback_explicit' AND column_name = 'app_id') THEN
        ALTER TABLE feedback_explicit ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinkt
    END IF;

    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'feedback_implicit' AND column_name = 'app_id') THEN
        ALTER TABLE feedback_implicit ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinkt
    END IF;

    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'feedback_voice' AND column_name = 'app_id') THEN
        ALTER TABLE feedback_voice ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank
    END IF;
END $$;

-- Execution Manifests
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'execution_manifests' AND column_name = 'app_id') THEN
        ALTER TABLE execution_manifests ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thin
    END IF;
END $$;

-- Think Tank-specific tables
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'thinktank_sessions' AND column_name = 'app_id') THEN
        ALTER TABLE thinktank_sessions ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'think
    END IF;

    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'thinktank_user_model_preferences' AND column_name =
```

```
        ALTER TABLE thinktank_user_model_preferences ADD COLUMN app_id VARCHAR(50) NOT NULL
    END IF;
END $$;

-- Collaboration
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'collaboration_rooms' AND column_name = 'app_id') THEN
        ALTER TABLE collaboration_rooms ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thin
    END IF;
END $$;

-- Time Machine snapshots
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name = 'chat_snapshots' AND column_name = 'app_id') THEN
        ALTER TABLE chat_snapshots ADD COLUMN app_id VARCHAR(50) NOT NULL DEFAULT 'thinktank
    END IF;
END $$;

-- ============================================================================
-- CREATE INDEXES FOR app_id COLUMNS
-- ============================================================================

-- Indexes for efficient app-scoped queries
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_thinktank_chats_app ON thinktank_chats(tenant_id
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_thinktank_messages_app ON thinktank_messages(ten
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_concurrent_sessions_app ON concurrent_sessions(t
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_voice_sessions_app ON voice_sessions(tenant_id,
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_memory_stores_app ON memory_stores(tenant_id, ap
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_canvases_app ON canvases(tenant_id, app_id);
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_user_personas_app ON user_personas(tenant_id, ap
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_user_preferences_app ON user_preferences(tenant_
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_feedback_explicit_app ON feedback_explicit(tenan
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_execution_manifests_app ON execution_manifests(t

-- ============================================================================
-- UPDATE RLS POLICIES TO INCLUDE app_id
-- ============================================================================

-- Drop existing policies and recreate with app_id filtering
-- Using a function to standardize policy creation

CREATE OR REPLACE FUNCTION create_app_isolation_policy(
```

```
    table_name TEXT,
    policy_name TEXT DEFAULT NULL
) RETURNS VOID AS $$
DECLARE
    p_name TEXT;
BEGIN
    p_name := COALESCE(policy_name, table_name || '_app_isolation');

    -- Drop existing policy if exists
    EXECUTE format('DROP POLICY IF EXISTS %I ON %I', p_name, table_name);

    -- Create new policy with app_id filtering
    EXECUTE format('
        CREATE POLICY %I ON %I
        FOR ALL
        USING (
            tenant_id = current_tenant_id()
            AND app_id = current_app_id()
        )
    ', p_name, table_name);
END;
$$ LANGUAGE plpgsql;

-- Apply app isolation policies to all relevant tables
SELECT create_app_isolation_policy('thinktank_chats');
SELECT create_app_isolation_policy('thinktank_messages');
SELECT create_app_isolation_policy('thinktank_conversations');
SELECT create_app_isolation_policy('concurrent_sessions');
SELECT create_app_isolation_policy('voice_sessions');
SELECT create_app_isolation_policy('memory_stores');
SELECT create_app_isolation_policy('memories');
SELECT create_app_isolation_policy('canvases');
SELECT create_app_isolation_policy('artifacts');
SELECT create_app_isolation_policy('user_personas');
SELECT create_app_isolation_policy('scheduled_prompts');
SELECT create_app_isolation_policy('user_preferences');
SELECT create_app_isolation_policy('user_memory');
SELECT create_app_isolation_policy('user_behavior_patterns');
SELECT create_app_isolation_policy('feedback_explicit');
SELECT create_app_isolation_policy('feedback_implicit');
SELECT create_app_isolation_policy('feedback_voice');
SELECT create_app_isolation_policy('execution_manifests');
SELECT create_app_isolation_policy('thinktank_sessions');
SELECT create_app_isolation_policy('thinktank_user_model_preferences');
SELECT create_app_isolation_policy('collaboration_rooms');
```

```sql
-- ============================================================================
-- ADMIN CROSS-APP VIEW POLICIES
-- ============================================================================
-- Platform administrators need to view data across apps for support

CREATE OR REPLACE FUNCTION create_admin_view_policy(
    table_name TEXT
) RETURNS VOID AS $$
BEGIN
    EXECUTE format('DROP POLICY IF EXISTS %I ON %I', table_name || '_admin_view', table_name

    EXECUTE format('
        CREATE POLICY %I ON %I
        FOR SELECT
        USING (
            tenant_id = current_tenant_id()
            AND current_setting(''app.is_admin'', true) = ''true''
        )
    ', table_name || '_admin_view', table_name);
END;
$$ LANGUAGE plpgsql;

-- Apply admin view policies
SELECT create_admin_view_policy('thinktank_chats');
SELECT create_admin_view_policy('thinktank_messages');
SELECT create_admin_view_policy('concurrent_sessions');
SELECT create_admin_view_policy('memory_stores');
SELECT create_admin_view_policy('canvases');
SELECT create_admin_view_policy('user_personas');
SELECT create_admin_view_policy('feedback_explicit');
SELECT create_admin_view_policy('execution_manifests');

-- ============================================================================
-- APP REGISTRY TABLE
-- ============================================================================
-- Track all registered applications

CREATE TABLE IF NOT EXISTS app_registry (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    display_name VARCHAR(200) NOT NULL,
    description TEXT,

    -- App type
    app_type VARCHAR(50) NOT NULL CHECK (app_type IN ('client_app', 'consumer', 'admin', 'in
```

```
    -- Cognito configuration
    cognito_user_pool_id VARCHAR(100),
    cognito_client_id VARCHAR(100),

    -- Routing
    subdomain VARCHAR(100),
    custom_domain VARCHAR(255),

    -- Features enabled for this app
    features JSONB DEFAULT '{}',

    -- Status
    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'inactive', 'de

    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

-- Seed default applications
INSERT INTO app_registry (id, name, display_name, description, app_type, subdomain) VALUES
    ('thinktank', 'thinktank', 'Think Tank', 'Consumer-facing AI chat application', 'consume
    ('thinktank', 'thinktank', 'Think Tank', 'Collaborative AI workspace', 'client_app', 'th
    ('launchboard', 'launchboard', 'Launch Board', 'Project management with AI', 'client_app
    ('alwaysme', 'alwaysme', 'Always Me', 'Personal AI assistant', 'client_app', 'alwaysme')
    ('mechanicalmaker', 'mechanicalmaker', 'Mechanical Maker', 'Engineering AI tools', 'clie
    ('admin', 'admin', 'Admin Dashboard', 'Platform administration', 'admin', 'admin')
ON CONFLICT (id) DO NOTHING;

-- ============================================================================
-- CROSS-APP CORRELATION TABLE (Admin Only)
-- ============================================================================
-- For platform admins to correlate user identities across apps
-- End users CANNOT access this table

CREATE TABLE IF NOT EXISTS cross_app_user_correlation (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Email is the correlation key
    email VARCHAR(255) NOT NULL,

    -- App user IDs for each app
    app_user_ids JSONB NOT NULL DEFAULT '{}',
    -- Example: {"thinktank": "uuid-1", "launchboard": "uuid-2", "launchboard": "uuid-3"}
```

```
    -- Metadata
    first_seen_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    last_activity_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

    -- Unique per tenant
    UNIQUE(tenant_id, email)
);

-- Only admins can access this table
ALTER TABLE cross_app_user_correlation ENABLE ROW LEVEL SECURITY;

CREATE POLICY cross_app_admin_only ON cross_app_user_correlation
    FOR ALL
    USING (
        tenant_id = current_tenant_id()
        AND current_setting('app.is_admin', true) = 'true'
    );

-- ============================================================================
-- MIGRATION TRACKING
-- ============================================================================

INSERT INTO schema_migrations (version, name, applied_by, checksum)
VALUES ('040', 'app_level_isolation', 'system', md5('v4.6.0_app_isolation'))
ON CONFLICT (version) DO NOTHING;
```

40.3 COGNITO CONFIGURATION

Per-App User Pool Stack

```
// packages/infrastructure/lib/stacks/app-cognito.stack.ts

import * as cdk from 'aws-cdk-lib';
import * as cognito from 'aws-cdk-lib/aws-cognito';
import { Construct } from 'constructs';

export interface AppCognitoStackProps extends cdk.StackProps {
  appId: string;
  appName: string;
  environment: string;
  domain: string;
  tier: number;
}

/**
```

```typescript
 * Creates a dedicated Cognito User Pool for each application.
 * This ensures complete identity isolation between apps.
 */
export class AppCognitoStack extends cdk.Stack {
  public readonly userPool: cognito.UserPool;
  public readonly userPoolClient: cognito.UserPoolClient;
  public readonly userPoolDomain: cognito.UserPoolDomain;

  constructor(scope: Construct, id: string, props: AppCognitoStackProps) {
    super(scope, id, props);

    const resourcePrefix = `${props.appId}-${props.environment}`;

    // ============================================================================
    // USER POOL (Per-App)
    // ============================================================================

    this.userPool = new cognito.UserPool(this, 'UserPool', {
      userPoolName: `${resourcePrefix}-users`,

      // Self sign-up enabled for Think Tank, disabled for client apps
      selfSignUpEnabled: props.appId === 'thinktank',

      // Sign-in options
      signInAliases: {
        email: true,
        username: false,
      },

      // Password policy
      passwordPolicy: {
        minLength: 12,
        requireLowercase: true,
        requireUppercase: true,
        requireDigits: true,
        requireSymbols: true,
      },

      // MFA configuration
      mfa: props.tier >= 3
        ? cognito.Mfa.OPTIONAL
        : cognito.Mfa.OFF,
      mfaSecondFactor: {
        sms: true,
        otp: true,
      },
```

```
    // Account recovery
    accountRecovery: cognito.AccountRecovery.EMAIL_ONLY,

    // Standard attributes
    standardAttributes: {
      email: {
        required: true,
        mutable: true,
      },
      fullname: {
        required: false,
        mutable: true,
      },
    },

    // Custom attributes - CRITICAL: app_id is immutable
    customAttributes: {
      tenantId: new cognito.StringAttribute({ mutable: false }),
      appId: new cognito.StringAttribute({ mutable: false }), // NEW: App identifier
      role: new cognito.StringAttribute({ mutable: true }),
      appUserId: new cognito.StringAttribute({ mutable: false }), // NEW: App-scoped user
    },

    // Lambda triggers for app isolation
    lambdaTriggers: {
      preSignUp: this.createPreSignUpLambda(resourcePrefix, props.appId),
      postConfirmation: this.createPostConfirmationLambda(resourcePrefix, props.appId),
      preTokenGeneration: this.createPreTokenGenerationLambda(resourcePrefix, props.appId)
    },

    // Removal policy
    removalPolicy: props.environment === 'prod'
      ? cdk.RemovalPolicy.RETAIN
      : cdk.RemovalPolicy.DESTROY,
  });

  // ============================================================================
  // USER POOL CLIENT (Per-App)
  // ============================================================================

  this.userPoolClient = this.userPool.addClient('UserPoolClient', {
    userPoolClientName: `${resourcePrefix}-web-client`,

    // OAuth configuration
    oAuth: {
```

```
      flows: {
        authorizationCodeGrant: true,
        implicitCodeGrant: false,
      },
      scopes: [
        cognito.OAuthScope.EMAIL,
        cognito.OAuthScope.OPENID,
        cognito.OAuthScope.PROFILE,
      ],
      callbackUrls: [
        `https://${props.appId}.${props.domain}/auth/callback`,
        'http://localhost:3000/auth/callback',
      ],
      logoutUrls: [
        `https://${props.appId}.${props.domain}/auth/logout`,
        'http://localhost:3000/auth/logout',
      ],
    },

    // Token configuration
    accessTokenValidity: cdk.Duration.hours(1),
    idTokenValidity: cdk.Duration.hours(1),
    refreshTokenValidity: cdk.Duration.days(30),

    // Auth flows
    authFlows: {
      userPassword: true,
      userSrp: true,
    },

    // Prevent user existence errors
    preventUserExistenceErrors: true,

    // Read/write attributes
    readAttributes: new cognito.ClientAttributes()
      .withStandardAttributes({
        email: true,
        fullname: true,
      })
      .withCustomAttributes('tenantId', 'appId', 'role', 'appUserId'),

    writeAttributes: new cognito.ClientAttributes()
      .withStandardAttributes({
        fullname: true,
      }),
  });
```

```
    // ============================================================================
    // USER POOL DOMAIN (Per-App)
    // ============================================================================

    this.userPoolDomain = this.userPool.addDomain('Domain', {
      cognitoDomain: {
        domainPrefix: `${props.appId}-${props.environment}-auth`,
      },
    });

    // ============================================================================
    // OUTPUTS
    // ============================================================================

    new cdk.CfnOutput(this, 'UserPoolId', {
      value: this.userPool.userPoolId,
      exportName: `${resourcePrefix}-user-pool-id`,
    });

    new cdk.CfnOutput(this, 'UserPoolClientId', {
      value: this.userPoolClient.userPoolClientId,
      exportName: `${resourcePrefix}-user-pool-client-id`,
    });

    new cdk.CfnOutput(this, 'UserPoolDomain', {
      value: this.userPoolDomain.domainName,
      exportName: `${resourcePrefix}-user-pool-domain`,
    });
  }

  /**
   * Pre-SignUp Lambda: Validates sign-up requests and enforces app isolation
   */
  private createPreSignUpLambda(resourcePrefix: string, appId: string): lambda.Function {
    return new lambdaNodejs.NodejsFunction(this, 'PreSignUpFunction', {
      functionName: `${resourcePrefix}-pre-signup`,
      runtime: lambda.Runtime.NODEJS_20_X,
      handler: 'handler',
      entry: 'lambda/cognito/pre-signup.ts',
      environment: {
        APP_ID: appId,
      },
    });
  }
```

```
  /**
   * Post-Confirmation Lambda: Creates app_users record after confirmation
   */
  private createPostConfirmationLambda(resourcePrefix: string, appId: string): lambda.Functi
    return new lambdaNodejs.NodejsFunction(this, 'PostConfirmationFunction', {
      functionName: `${resourcePrefix}-post-confirmation`,
      runtime: lambda.Runtime.NODEJS_20_X,
      handler: 'handler',
      entry: 'lambda/cognito/post-confirmation.ts',
      environment: {
        APP_ID: appId,
      },
    });
  }

  /**
   * Pre-Token-Generation Lambda: Adds app_id and app_user_id to JWT claims
   */
  private createPreTokenGenerationLambda(resourcePrefix: string, appId: string): lambda.Func
    return new lambdaNodejs.NodejsFunction(this, 'PreTokenGenerationFunction', {
      functionName: `${resourcePrefix}-pre-token-gen`,
      runtime: lambda.Runtime.NODEJS_20_X,
      handler: 'handler',
      entry: 'lambda/cognito/pre-token-generation.ts',
      environment: {
        APP_ID: appId,
      },
    });
  }
}
```

Cognito Lambda Triggers

```
// lambda/cognito/pre-signup.ts

import { PreSignUpTriggerEvent, PreSignUpTriggerHandler } from 'aws-lambda';

/**
 * Pre-SignUp Lambda
 * Validates that the user is signing up for the correct app
 */
export const handler: PreSignUpTriggerHandler = async (event) => {
  const appId = process.env.APP_ID;

  // For Think Tank, auto-confirm email in non-prod environments
  if (appId === 'thinktank' && process.env.ENVIRONMENT !== 'prod') {
```

```
    event.response.autoConfirmUser = true;
    event.response.autoVerifyEmail = true;
  }

  // Log sign-up attempt for audit
  console.log('Pre-SignUp', {
    appId,
    email: event.request.userAttributes.email,
    userPoolId: event.userPoolId,
  });

  return event;
};
```

```
// lambda/cognito/post-confirmation.ts

import { PostConfirmationTriggerEvent, PostConfirmationTriggerHandler } from 'aws-lambda';
import { getDbClient } from '../shared/db';

/**
 * Post-Confirmation Lambda
 * Creates the app_users record after user confirms their email
 */
export const handler: PostConfirmationTriggerHandler = async (event) => {
  const appId = process.env.APP_ID!;
  const tenantId = event.request.userAttributes['custom:tenantId'];
  const cognitoSub = event.request.userAttributes.sub;
  const email = event.request.userAttributes.email;
  const fullName = event.request.userAttributes.name || '';

  const db = await getDbClient();

  try {
    // Create app-scoped user record
    const result = await db.query(`
      INSERT INTO app_users (
        tenant_id, app_id, cognito_sub, email,
        first_name, last_name, display_name, status
      )
      VALUES ($1, $2, $3, $4, $5, $6, $7, 'active')
      ON CONFLICT (tenant_id, app_id, email) DO UPDATE SET
        cognito_sub = EXCLUDED.cognito_sub,
        last_login_at = NOW(),
        login_count = app_users.login_count + 1
      RETURNING id
    `, [
```

```
      tenantId,
      appId,
      cognitoSub,
      email,
      fullName.split(' ')[0] || '',
      fullName.split(' ').slice(1).join(' ') || '',
      fullName || email.split('@')[0],
    ]);

    const appUserId = result.rows[0].id;

    // Update cross-app correlation (for admin use only)
    await db.query(`
      INSERT INTO cross_app_user_correlation (tenant_id, email, app_user_ids)
      VALUES ($1, $2, $3::jsonb)
      ON CONFLICT (tenant_id, email) DO UPDATE SET
        app_user_ids = cross_app_user_correlation.app_user_ids || $3::jsonb,
        last_activity_at = NOW()
    `, [tenantId, email, JSON.stringify({ [appId]: appUserId })]);

    console.log('Created app_user', { appId, appUserId, email });

  } catch (error) {
    console.error('Error creating app_user', error);
    throw error;
  }

  return event;
};

// lambda/cognito/pre-token-generation.ts

import { PreTokenGenerationTriggerEvent, PreTokenGenerationTriggerHandler } from 'aws-lambda
import { getDbClient } from '../shared/db';

/**
 * Pre-Token-Generation Lambda
 * Adds app_id and app_user_id to JWT claims
 */
export const handler: PreTokenGenerationTriggerHandler = async (event) => {
  const appId = process.env.APP_ID!;
  const cognitoSub = event.request.userAttributes.sub;

  const db = await getDbClient();

  // Get app_user_id for this user
```

```
  const result = await db.query(`
    SELECT id FROM app_users
    WHERE cognito_sub = $1 AND app_id = $2
  `, [cognitoSub, appId]);

  const appUserId = result.rows[0]?.id;

  if (!appUserId) {
    console.error('No app_user found for', { cognitoSub, appId });
    throw new Error('User not found in this application');
  }

  // Add custom claims to the ID token
  event.response.claimsOverrideDetails = {
    claimsToAddOrOverride: {
      'custom:app_id': appId,
      'custom:app_user_id': appUserId,
    },
  };

  return event;
};
```

40.4 LAMBDA AUTH CONTEXT UPDATE

> **NOTE**: The `AuthContext` interface and `extractAuthContext` function are defined in **Section 4** (`lambda/shared/auth.ts`). The canonical definition already includes all app isolation fields: `appId`, `appUserId`, `isSuperAdmin`, `tokenExpiry`.
>
> **DO NOT redefine these types.** Import from `@radiant/shared` or `../shared/auth`.

App Isolation Validation Logic

The following validation logic is already integrated into Section 4's `extractAuthContext`:

```
// Already in Section 4: lambda/shared/auth.ts
// Key validation points for app isolation:

// 1. Extract app-specific claims
const appId = claims['custom:appId'] || claims['custom:app_id'];
const appUserId = claims['custom:appUserId'] || claims['custom:app_user_id'];

// 2. Validate required claims for app isolation
```

```
if (!appId) throw new UnauthorizedError('Missing app ID');
if (!appUserId) throw new UnauthorizedError('Missing app user ID');

// 3. Validate app_id matches route (defense in depth)
const routeAppId = extractAppIdFromRoute(event);
if (routeAppId && routeAppId !== appId) {
  throw new ForbiddenError(`Token app_id (${appId}) does not match route (${routeAppId})`);
}
```

Extract App ID from Route Helper

```
// lambda/shared/auth.ts - extractAppIdFromRoute function
/**
 * Extract app_id from route for validation
 */
function extractAppIdFromRoute(event: APIGatewayProxyEvent): string | null {
  // Extract from subdomain: thinktank.domain.com -> thinktank
  const host = event.headers.Host || event.headers.host;
  if (host) {
    const subdomain = host.split('.')[0];
    if (['thinktank', 'launchboard', 'alwaysme', 'mechanicalmaker'].includes(subdomain)) {
      return subdomain;
    }
  }

  // Extract from path: /api/thinktank/... -> thinktank
  const pathMatch = event.path.match(/^\/api\/(thinktank|launchboard|alwaysme|mechanicalmake
  if (pathMatch) {
    return pathMatch[1];
  }

  return null;
}

/**
 * Set database context for RLS policies
 * CRITICAL: This must be called before any database queries
 */
export async function setDatabaseContext(
  db: PoolClient,
  auth: AuthContext
): Promise<void> {
  await db.query(`
    SET LOCAL app.current_tenant_id = $1;
    SET LOCAL app.current_app_id = $2;
    SET LOCAL app.is_admin = $3;
```

```
  `, [auth.tenantId, auth.appId, auth.isAdmin ? 'true' : 'false']);
}

/**
 * Create authenticated database client with context
 */
export async function getAuthenticatedDb(
  auth: AuthContext
): Promise<{ client: PoolClient; release: () => void }> {
  const pool = await getPool();
  const client = await pool.connect();

  try {
    await setDatabaseContext(client, auth);

    return {
      client,
      release: () => client.release(),
    };
  } catch (error) {
    client.release();
    throw error;
  }
}
```

40.5 API ROUTING BY APP

API Gateway Per-App Routes

```
// packages/infrastructure/lib/stacks/api.stack.ts - Addition

/**
 * Create per-app API routes with app_id validation
 */
private createAppRoutes(props: APIStackProps): void {
  const apps = ['thinktank', 'launchboard', 'alwaysme', 'mechanicalmaker'];

  for (const appId of apps) {
    // Create resource for this app
    const appResource = this.api.root.addResource(appId);

    // Apply app-specific authorizer
    const authorizer = new apigateway.CognitoUserPoolsAuthorizer(
      this,
      `${appId}Authorizer`,
```

```
  {
    cognitoUserPools: [props.appUserPools[appId]],
    identitySource: 'method.request.header.Authorization',
  }
);

// Chat endpoint
const chatsResource = appResource.addResource('chats');
chatsResource.addMethod('GET',
  new apigateway.LambdaIntegration(this.chatFunction),
  { authorizer }
);
chatsResource.addMethod('POST',
  new apigateway.LambdaIntegration(this.chatFunction),
  { authorizer }
);

// Chat by ID
const chatResource = chatsResource.addResource('{chatId}');
chatResource.addMethod('GET',
  new apigateway.LambdaIntegration(this.chatFunction),
  { authorizer }
);
chatResource.addMethod('DELETE',
  new apigateway.LambdaIntegration(this.chatFunction),
  { authorizer }
);

// Messages
const messagesResource = chatResource.addResource('messages');
messagesResource.addMethod('GET',
  new apigateway.LambdaIntegration(this.chatFunction),
  { authorizer }
);
messagesResource.addMethod('POST',
  new apigateway.LambdaIntegration(this.chatFunction),
  { authorizer }
);

// User preferences (app-specific)
const preferencesResource = appResource.addResource('preferences');
preferencesResource.addMethod('GET',
  new apigateway.LambdaIntegration(this.preferencesFunction),
  { authorizer }
);
preferencesResource.addMethod('PUT',
```

```
      new apigateway.LambdaIntegration(this.preferencesFunction),
      { authorizer }
    );

    // Memory (app-specific)
    const memoryResource = appResource.addResource('memory');
    memoryResource.addMethod('GET',
      new apigateway.LambdaIntegration(this.memoryFunction),
      { authorizer }
    );
  }
}
```

40.6 ADMIN DASHBOARD UPDATES

Cross-App User View (Admin Only)

```
// apps/admin-dashboard/src/app/(dashboard)/users/cross-app/page.tsx

'use client';

import { useState } from 'react';
import { useQuery } from '@tanstack/react-query';
import {
  Box,
  Card,
  CardContent,
  Typography,
  Table,
  TableHead,
  TableBody,
  TableRow,
  TableCell,
  Chip,
  TextField,
  InputAdornment,
  IconButton,
  Tooltip,
  Dialog,
  DialogTitle,
  DialogContent,
} from '@mui/material';
import { Search, Visibility, Apps, Person } from '@mui/icons-material';

interface CrossAppUser {
```

```
  email: string;
  tenantId: string;
  appUsers: {
    appId: string;
    appUserId: string;
    displayName: string;
    lastLoginAt: string;
    status: string;
    chatCount: number;
  }[];
  firstSeenAt: string;
  lastActivityAt: string;
}

export default function CrossAppUsersPage() {
  const [search, setSearch] = useState('');
  const [selectedUser, setSelectedUser] = useState<CrossAppUser | null>(null);

  const { data: users, isLoading } = useQuery({
    queryKey: ['cross-app-users', search],
    queryFn: async () => {
      const response = await fetch(
        `/api/admin/users/cross-app?search=${encodeURIComponent(search)}`
      );
      return response.json() as Promise<CrossAppUser[]>;
    },
  });

  const appColors: Record<string, string> = {
    thinktank: 'primary',
    thinktank: 'secondary',
    launchboard: 'success',
    alwaysme: 'warning',
    mechanicalmaker: 'info',
  };

  return (
    <Box>
      <Typography variant="h4" gutterBottom>
        Cross-App User Correlation
      </Typography>
      <Typography variant="body2" color="textSecondary" sx={{ mb: 3 }}>
        View users across all applications. This is for admin support purposes only.
        End users cannot see data from other applications.
      </Typography>
```

```
<Card sx={{ mb: 3 }}>
  <CardContent>
    <TextField
      fullWidth
      placeholder="Search by email..."
      value={search}
      onChange={(e) => setSearch(e.target.value)}
      InputProps={{
        startAdornment: (
          <InputAdornment position="start">
            <Search />
          </InputAdornment>
        ),
      }}
    />
  </CardContent>
</Card>

<Card>
  <Table>
    <TableHead>
      <TableRow>
        <TableCell>Email</TableCell>
        <TableCell>Applications</TableCell>
        <TableCell>First Seen</TableCell>
        <TableCell>Last Activity</TableCell>
        <TableCell>Actions</TableCell>
      </TableRow>
    </TableHead>
    <TableBody>
      {users?.map((user) => (
        <TableRow key={user.email}>
          <TableCell>
            <Box sx={{ display: 'flex', alignItems: 'center', gap: 1 }}>
              <Person fontSize="small" />
              {user.email}
            </Box>
          </TableCell>
          <TableCell>
            <Box sx={{ display: 'flex', gap: 0.5, flexWrap: 'wrap' }}>
              {user.appUsers.map((appUser) => (
                <Chip
                  key={appUser.appId}
                  label={appUser.appId}
                  size="small"
                  color={appColors[appUser.appId] as any || 'default'}
```

```
                    variant={appUser.status === 'active' ? 'filled' : 'outlined'}
                  />
                ))}
              </Box>
            </TableCell>
            <TableCell>
              {new Date(user.firstSeenAt).toLocaleDateString()}
            </TableCell>
            <TableCell>
              {new Date(user.lastActivityAt).toLocaleString()}
            </TableCell>
            <TableCell>
              <Tooltip title="View Details">
                <IconButton onClick={() => setSelectedUser(user)}>
                  <Visibility />
                </IconButton>
              </Tooltip>
            </TableCell>
          </TableRow>
        ))}
      </TableBody>
    </Table>
  </Card>

  {/* User Detail Dialog */}
  <Dialog
    open={!!selectedUser}
    onClose={() => setSelectedUser(null)}
    maxWidth="md"
    fullWidth
  >
    <DialogTitle>
      <Box sx={{ display: 'flex', alignItems: 'center', gap: 1 }}>
        <Apps />
        User Details: {selectedUser?.email}
      </Box>
    </DialogTitle>
    <DialogContent>
      {selectedUser && (
        <Table>
          <TableHead>
            <TableRow>
              <TableCell>Application</TableCell>
              <TableCell>Display Name</TableCell>
              <TableCell>Status</TableCell>
              <TableCell>Chats</TableCell>
```

```
                    <TableCell>Last Login</TableCell>
                  </TableRow>
                </TableHead>
                <TableBody>
                  {selectedUser.appUsers.map((appUser) => (
                    <TableRow key={appUser.appId}>
                      <TableCell>
                        <Chip
                          label={appUser.appId}
                          color={appColors[appUser.appId] as any || 'default'}
                        />
                      </TableCell>
                      <TableCell>{appUser.displayName}</TableCell>
                      <TableCell>
                        <Chip
                          label={appUser.status}
                          size="small"
                          color={appUser.status === 'active' ? 'success' : 'default'}
                        />
                      </TableCell>
                      <TableCell>{appUser.chatCount}</TableCell>
                      <TableCell>
                        {appUser.lastLoginAt
                          ? new Date(appUser.lastLoginAt).toLocaleString()
                          : 'Never'}
                      </TableCell>
                    </TableRow>
                  ))}
                </TableBody>
              </Table>
            )}
          </DialogContent>
        </Dialog>
      </Box>
    );
}
```

40.7 MIGRATION GUIDE

Step-by-Step Migration for Existing Deployments

```
# Migration Guide: v4.5.0 → v4.6.0 (Application-Level Isolation)

## Overview
```

```
This migration adds app_id isolation to all user-facing tables. Existing data
will be migrated with a default app_id based on the primary application.

## Pre-Migration Checklist

- [ ] Backup database
- [ ] Schedule maintenance window (migration takes ~10-30 minutes depending on data size)
- [ ] Notify users of maintenance
- [ ] Verify AWS Lambda permissions for Cognito triggers

## Migration Steps

### 1. Apply Database Migration

```bash
# Apply the new migration
cd packages/infrastructure
pnpm run migrate:up

# Verify migration applied
psql $DATABASE_URL -c "SELECT * FROM schema_migrations WHERE version = '040';"
```

2. Migrate Existing Users to app_users

```
-- Migration script to copy existing users to app_users
-- Run this ONCE after applying 040_app_level_isolation.sql

INSERT INTO app_users (
    tenant_id,
    app_id,
    base_user_id,
    cognito_sub,
    email,
    email_verified,
    first_name,
    last_name,
    display_name,
    avatar_url,
    preferences,
    timezone,
    locale,
    status,
    last_login_at,
    created_at,
    updated_at
)
```
```

```sql
SELECT
    u.tenant_id,
    t.app_id,  -- Use tenant's app_id
    u.id,      -- Link to base user
    u.cognito_sub,
    u.email,
    u.email_verified,
    u.first_name,
    u.last_name,
    u.display_name,
    u.avatar_url,
    u.preferences,
    u.timezone,
    u.locale,
    u.status,
    u.last_login_at,
    u.created_at,
    u.updated_at
FROM users u
JOIN tenants t ON u.tenant_id = t.id
WHERE NOT EXISTS (
    SELECT 1 FROM app_users au
    WHERE au.tenant_id = u.tenant_id
    AND au.app_id = t.app_id
    AND au.email = u.email
);
```

3. Update Existing Data with app_id

```sql
-- Set app_id on existing data based on tenant's primary app
UPDATE thinktank_chats SET app_id = 'thinktank' WHERE app_id IS NULL OR app_id = '';
UPDATE thinktank_messages SET app_id = 'thinktank' WHERE app_id IS NULL OR app_id = '';
UPDATE concurrent_sessions SET app_id = 'thinktank' WHERE app_id IS NULL OR app_id = '';
-- ... repeat for all tables
```

4. Deploy New Cognito Pools

```bash
# Deploy per-app Cognito pools
cd packages/infrastructure
cdk deploy AppCognitoThinkTankStack
cdk deploy AppCognitoThinkTankStack
cdk deploy AppCognitoLaunchBoardStack
cdk deploy AppCognitoAlwaysMeStack
cdk deploy AppCognitoMechanicalMakerStack
```

5. Update API Gateway

```
# Deploy updated API with per-app routes
cdk deploy APIStack
```

6. Verify Migration

```
-- Verify app_users populated
SELECT app_id, COUNT(*) FROM app_users GROUP BY app_id;

-- Verify RLS policies working
SET app.current_tenant_id = 'your-tenant-id';
SET app.current_app_id = 'thinktank';
SELECT COUNT(*) FROM thinktank_chats; -- Should only show Think Tank chats

SET app.current_app_id = 'thinktank';
SELECT COUNT(*) FROM thinktank_chats; -- Should show 0 (no Think Tank chats in thinktank_cha
```

Rollback Procedure

If issues occur, you can temporarily disable app isolation:

```
-- Emergency rollback: Remove app_id from RLS policies
CREATE OR REPLACE FUNCTION current_app_id() RETURNS VARCHAR(50) AS $$
BEGIN
    RETURN NULL;  -- Returns NULL, effectively disabling app filtering
END;
$$ LANGUAGE plpgsql SECURITY DEFINER STABLE;
```

Post-Migration

- ☐ Monitor error rates in CloudWatch
- ☐ Verify cross-app isolation working (test with same email in different apps)
- ☐ Update client applications with new Cognito pool IDs
- ☐ Train support staff on cross-app user correlation dashboard

---

40.8 TESTING

Isolation Test Suite

```typescript
// tests/isolation/app-isolation.test.ts

import { describe, it, expect, beforeAll, afterAll } from '@jest/globals';
import { getDbClient, closePool } from '../utils/db';
```

```
describe('Application-Level Data Isolation', () => {
  let db: any;

  beforeAll(async () => {
    db = await getDbClient();
  });

  afterAll(async () => {
    await closePool();
  });

  describe('RLS Policies', () => {
    it('should isolate data between apps in same tenant', async () => {
      const tenantId = 'test-tenant-001';

      // Create test chats in different apps
      await db.query(`SET app.current_tenant_id = $1`, [tenantId]);

      // Insert chat in Think Tank
      await db.query(`SET app.current_app_id = 'thinktank'`);
      await db.query(`
        INSERT INTO thinktank_chats (id, tenant_id, app_id, user_id, title)
        VALUES ('chat-thinktank-1', $1, 'thinktank', 'user-1', 'Think Tank Chat')
      `, [tenantId]);

      // Insert chat in Think Tank
      await db.query(`SET app.current_app_id = 'thinktank'`);
      await db.query(`
        INSERT INTO thinktank_chats (id, tenant_id, app_id, user_id, title)
        VALUES ('chat-tt-1', $1, 'thinktank', 'user-1', 'Think Tank Chat')
      `, [tenantId]);

      // Query from Think Tank context
      await db.query(`SET app.current_app_id = 'thinktank'`);
      const thinktankChats = await db.query(`SELECT * FROM thinktank_chats`);

      expect(thinktankChats.rows).toHaveLength(1);
      expect(thinktankChats.rows[0].title).toBe('Think Tank Chat');

      // Query from Think Tank context
      await db.query(`SET app.current_app_id = 'thinktank'`);
      const ttChats = await db.query(`SELECT * FROM thinktank_chats`);

      expect(ttChats.rows).toHaveLength(1);
      expect(ttChats.rows[0].title).toBe('Think Tank Chat');
```

```
  });

  it('should allow admin cross-app view', async () => {
    const tenantId = 'test-tenant-001';

    await db.query(`SET app.current_tenant_id = $1`, [tenantId]);
    await db.query(`SET app.is_admin = 'true'`);

    // Admin should see all chats
    const allChats = await db.query(`
      SELECT * FROM thinktank_chats WHERE tenant_id = $1
    `, [tenantId]);

    expect(allChats.rows.length).toBeGreaterThanOrEqual(2);
  });

  it('should prevent cross-tenant access', async () => {
    await db.query(`SET app.current_tenant_id = 'other-tenant'`);
    await db.query(`SET app.current_app_id = 'thinktank'`);

    const otherTenantChats = await db.query(`SELECT * FROM thinktank_chats`);

    expect(otherTenantChats.rows).toHaveLength(0);
  });
});

describe('App Users', () => {
  it('should create separate app_users for same email', async () => {
    const tenantId = 'test-tenant-002';
    const email = 'alice@test.com';

    // Create app_user in Think Tank
    await db.query(`
      INSERT INTO app_users (tenant_id, app_id, cognito_sub, email, status)
      VALUES ($1, 'thinktank', 'sub-thinktank', $2, 'active')
    `, [tenantId, email]);

    // Create app_user in Think Tank
    await db.query(`
      INSERT INTO app_users (tenant_id, app_id, cognito_sub, email, status)
      VALUES ($1, 'thinktank', 'sub-thinktank', $2, 'active')
    `, [tenantId, email]);

    // Verify both exist
    const thinktankUser = await db.query(`
      SELECT * FROM app_users
```

```
        WHERE tenant_id = $1 AND app_id = 'thinktank' AND email = $2
      `, [tenantId, email]);

      const ttUser = await db.query(`
        SELECT * FROM app_users
        WHERE tenant_id = $1 AND app_id = 'thinktank' AND email = $2
      `, [tenantId, email]);

      expect(thinktankUser.rows).toHaveLength(1);
      expect(ttUser.rows).toHaveLength(1);
      expect(thinktankUser.rows[0].id).not.toBe(ttUser.rows[0].id);
    });
  });
});
```

40.9 SUMMARY

What v4.6.0 Achieves

Before (v4.5.0)	After (v4.6.0)
Users isolated by tenant_id only	Users isolated by tenant_id + app_id
Same email = same user across all apps	Same email = separate identity per app
Think Tank data visible to client apps	Think Tank completely isolated
Single Cognito pool per tenant	Separate Cognito pool per app
RLS filters by tenant only	RLS filters by tenant AND app
No cross-app admin view	Admin dashboard for cross-app correlation

Key Files Changed/Added

```
packages/
  infrastructure/
    lib/stacks/
      app-cognito.stack.ts          # NEW: Per-app Cognito
    migrations/
      040_app_level_isolation.sql    # NEW: Database changes
  lambda/
    cognito/
      pre-signup.ts                  # NEW: Sign-up validation
      post-confirmation.ts           # NEW: App user creation
      pre-token-generation.ts        # NEW: Add app claims
```

```
      shared/
          auth/
              context.ts                  # UPDATED: App context
apps/
   admin-dashboard/
      src/app/(dashboard)/users/
          cross-app/page.tsx              # NEW: Admin view
tests/
   isolation/
       app-isolation.test.ts              # NEW: Isolation tests
```

ðŸš€ HOW TO USE THIS PROMPT

This is the **definitive, fully deduplicated** implementation prompt for RADIANT v2.2.0. All duplicate type definitions have been removed and replaced with a single source of truth.

For Windsurf IDE:

1. Open Windsurf IDE
2. Create a new folder: `radiant`
3. Paste this entire prompt into Cascade (Claude Opus 4.5)
4. The AI will implement systematically, section by section
5. Review and commit incrementally

Implementation Order (Dependency Graph):

```
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€
â",  SECTION 0: Shared Types (@radiant/shared)                       â",
â",  â""â"€â"€ Must be implemented FIRST - all other sections import it  â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€
                              â",
                              â-¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€
â",  SECTION 1: Foundation & Swift App                               â",
â",  â""â"€â"€ Depends on: Section 0                                         â",
â",  â""â"€â"€ Creates: Monorepo structure, Swift macOS app                  â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€
                              â",
                              â-¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€
â",  SECTION 2: CDK Infrastructure                                    â",
â",  â""â"€â"€ Depends on: Sections 0, 1                                     â",
â",  â""â"€â"€ Creates: VPC, Aurora, DynamoDB, S3, KMS, WAF                  â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€
```

```
                              â",
                              â–¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
â",  SECTION 3: CDK AI & API Stacks                                 â",
â",  â""â"€â"€ Depends on: Sections 0-2                                    â",
â",  â""â"€â"€ Creates: Cognito, LiteLLM, API Gateway, AppSync             â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
                              â",
                              â–¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
â",  SECTION 4: Lambda Functions - Core                              â",
â",  â""â"€â"€ Depends on: Sections 0-3                                    â",
â",  â""â"€â"€ Creates: Router, Chat, Models, Providers, PHI handlers      â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
                              â",
                              â–¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
â",  SECTION 5: Lambda Functions - Admin & Billing                   â",
â",  â""â"€â"€ Depends on: Sections 0-4                                    â",
â",  â""â"€â"€ Creates: Invitations, Approvals, Metering, Billing          â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
                              â",
                              â–¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
â",  SECTION 6: Self-Hosted Models & Services                        â",
â",  â""â"€â"€ Depends on: Sections 0-5                                    â",
â",  â""â"€â"€ Creates: 30+ SageMaker models, Thermal state management    â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
                              â",
                              â–¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
â",  SECTION 7: Database Schema (CANONICAL)                          â",
â",  â""â"€â"€ Depends on: Sections 0-6                                    â",
â",  â""â"€â"€ Creates: Complete PostgreSQL schema, all migrations        â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
                              â",
                              â–¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
â",  SECTION 8: Admin Web Dashboard                                  â",
â",  â""â"€â"€ Depends on: Sections 0-7                                    â",
â",  â""â"€â"€ Creates: Next.js 14 dashboard, all management UIs          â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
                              â",
                              â–¼
â"Œâ"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â
â",  SECTION 9: Assembly & Deployment                                â",
```

```
â",  â""â"€â"€ Depends on: ALL sections                                  â",
â",  â""â"€â"€ Provides: Verification, testing, troubleshooting          â",
â""â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€â"€
```

ðŸ"‹ CONFIGURATION - REPLACE THESE PLACEHOLDERS

Before deployment, find and replace these placeholders:

Placeholder	Description	Your Value
‘YOUR_DOMAIN.com‘	Your base domain	e.g., ‘zynapses.com‘
‘YOUR_APP_ID‘	Application identifier	e.g., ‘thinktank‘
‘YOUR_AWS_ACCOUNT_ID‘	12-digit AWS account	e.g., ‘123456789012‘
‘YOUR_AWS_REGION‘	Primary AWS region	e.g., ‘us-east-1‘
‘YOUR_ORG_IDENTIFIER‘	Bundle ID prefix	e.g., ‘com.yourcompany‘

ðŸ"Š CANONICAL DATABASE TABLES

Use these table names consistently (not the alternatives):

Canonical Name	Do NOT Use
`tenants`	-
`users`	-
`administrators`	`admin_users`
`invitations`	`admin_invitations`
`approval_requests`	`promotions`, `admin_approvals`
`providers`	-
`models`	`ai_models` (legacy)
`ai_models`	- (v4.1.0 orchestration registry)
`model_licenses`	- (v4.1.0 license tracking)
`model_dependencies`	- (v4.1.0 dependency tracking)
`workflow_definitions`	- (v4.1.0 workflow DAGs)
`workflow_tasks`	- (v4.1.0 workflow steps)
`workflow_parameters`	- (v4.1.0 configurable params)
`workflow_executions`	- (v4.1.0 execution tracking)
`task_executions`	- (v4.1.0 task-level logs)
`service_definitions`	- (v4.1.0 mid-level services)
`orchestration_audit_log`	- (v4.1.0 change audit)

Canonical Name	Do NOT Use
unified_model_registry	- (v4.2.0 combined view - ALL models)
registry_sync_log	- (v4.2.0 sync history)
provider_health	- (v4.2.0 health monitoring)
self_hosted_models	- (v4.2.0 SageMaker models catalog)
execution_manifests	- (v4.3.0 full execution provenance)
feedback_explicit	- (v4.3.0 thumbs up/down + categories)
feedback_implicit	- (v4.3.0 regenerate, copy, abandon signals)
feedback_voice	- (v4.3.0 voice feedback with transcription)
neural_model_scores	- (v4.3.0 learned model effectiveness)
neural_routing_recommendations	- (v4.3.0 Brain advice from Neural Engine)
user_trust_scores	- (v4.3.0 anti-gaming trust levels)
ab_experiments	- (v4.3.0 A/B testing experiments)
ab_experiment_assignments	- (v4.3.0 user experiment assignments)
localization_languages	- (v4.7.0 supported languages)
localization_registry	- (v4.7.0 all translatable strings)
localization_translations	- (v4.7.0 per-language translations)
localization_audit_log	- (v4.7.0 translation change history)
configuration_categories	- (v4.8.0 config categories)
system_configuration	- (v4.8.0 global config parameters)
tenant_configuration_overrides	- (v4.8.0 per-tenant config overrides)
configuration_audit_log	- (v4.8.0 config change history)
configuration_cache_invalidation	- (v4.8.0 cache invalidation queue)
usage_events	usage_records
invoices	-
audit_logs	-

SECTION 41: COMPLETE INTERNATIONALIZATION SYSTEM (v4.7.0)

CRITICAL: This section implements complete i18n/localization for RADIANT. ZERO hardcoded strings allowed - ALL user-facing text must come from the localization registry.

41.1 ARCHITECTURE OVERVIEW

The Problem

Before v4.7.0, RADIANT had scattered hardcoded strings throughout the codebase:

```
BEFORE (v4.6.0 and earlier):

 Hardcoded Strings Everywhere


  React Component:
  <Button>Submit</Button>  ← Hardcoded!
  <Alert>Error occurred</Alert>  ← Hardcoded!

  Lambda Response:
  { message: "User not found" }  ← Hardcoded!
  throw new Error("Invalid input")  ← Hardcoded!

  Swift App:
  Text("Welcome")  ← Hardcoded!
  Alert("Connection failed")  ← Hardcoded!

  Problems:
     Cannot support multiple languages
     No central place to update text
     Inconsistent terminology across apps
     No way to A/B test copy changes
```

The Solution

v4.7.0 implements **complete database-driven localization**:

```
AFTER (v4.7.0):

  Centralized Localization Registry



    localization_registry (Single Source of Truth)

      key: "button.submit"
      default_text: "Submit"
      context: "Primary action button"
      category: "ui.buttons"




    localization_translations
       en: "Submit"           (status: approved)
       es: "Enviar"           (status: approved)
       fr: "Soumettre"        (status: approved)
       de: "Absenden"         (status: ai_translated)
       ja: " "              (status: approved)
       ... 18 languages total


  React: {t('button.submit')}
  Lambda: t('error.user_not_found', { userId })
  Swift: L10n.button.submit
```

Supported Languages (18)

Code	Language	Native Name	RTL	Status
`en`	English	English	No	Primary
`es`	Spanish	Español	No	Supported
`fr`	French	Français	No	Supported
`de`	German	Deutsch	No	Supported
`pt`	Portuguese	Português	No	Supported
`it`	Italian	Italiano	No	Supported
`nl`	Dutch	Nederlands	No	Supported

Code	Language	Native Name	RTL	Status
pl	Polish	Polski	No	Supported
ru	Russian		No	Supported
tr	Turkish	Türkçe	No	Supported
ja	Japanese		No	Supported
ko	Korean		No	Supported
zh-CN	Chinese (Simplified)		No	Supported
zh-TW	Chinese (Traditional)		No	Supported
ar	Arabic		**Yes**	Supported
hi	Hindi		No	Supported
th	Thai		No	Supported
vi	Vietnamese	Tiếng Việt	No	Supported

System Architecture

```
                    RADIANT v4.7.0 LOCALIZATION ARCHITECTURE


  BUILD TIME (Prevention)

    ESLint Plugin: @radiant/eslint-plugin-i18n

       BLOCKS: <Button>Submit</Button>
       BLOCKS: throw new Error("Invalid input")
       BLOCKS: { message: "User not found" }
       ALLOWS: <Button>{t('button.submit')}</Button>
       ALLOWS: throw new LocalizedError('error.invalid_input')




  RUNTIME (Database-Driven)

    PostgreSQL: localization_registry + localization_translations
      All strings registered with unique keys
      Translations for 18 languages
      Status tracking (approved, ai_translated, needs_review)
      Version history for all changes



  AUTO-TRANSLATION (AI-Powered)

    Lambda: localization-translate
```

```
      Triggered when new registry entry added
      Uses AWS Bedrock (Claude) for translation
      Sets status = 'ai_translated' (NOT approved)
      Sends notification to admin for review


  ADMIN REVIEW (Human Approval)

    Admin Dashboard: /admin/localization
      View all strings needing review
      Edit translations inline
      Approve AI translations
      Translation coverage dashboard
      Bulk operations (approve all, export, import)
```

41.2 DATABASE SCHEMA

Migration: 041_localization_system.sql

```sql
-- ============================================================================
-- RADIANT v4.7.0 - Complete Internationalization System
-- Migration: 041_localization_system.sql
-- ============================================================================

-- ============================================================================
-- 41.2.1 Supported Languages Table
-- ============================================================================

CREATE TABLE localization_languages (
    code VARCHAR(10) PRIMARY KEY,  -- ISO 639-1 + region (e.g., 'en', 'zh-CN')
    name VARCHAR(100) NOT NULL,     -- English name
    native_name VARCHAR(100) NOT NULL,  -- Name in native script
    is_rtl BOOLEAN DEFAULT false,   -- Right-to-left language
    is_active BOOLEAN DEFAULT true, -- Available for selection
    display_order INTEGER DEFAULT 100,

    -- Metadata
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);
```

```sql
-- Insert supported languages
INSERT INTO localization_languages (code, name, native_name, is_rtl, display_order) VALUES
('en', 'English', 'English', false, 1),
('es', 'Spanish', 'Español', false, 2),
('fr', 'French', 'Français', false, 3),
('de', 'German', 'Deutsch', false, 4),
('pt', 'Portuguese', 'Português', false, 5),
('it', 'Italian', 'Italiano', false, 6),
('nl', 'Dutch', 'Nederlands', false, 7),
('pl', 'Polish', 'Polski', false, 8),
('ru', 'Russian', '    ', false, 9),
('tr', 'Turkish', 'Türkçe', false, 10),
('ja', 'Japanese', '  ', false, 11),
('ko', 'Korean', '  ', false, 12),
('zh-CN', 'Chinese (Simplified)', '  ', false, 13),
('zh-TW', 'Chinese (Traditional)', '  ', false, 14),
('ar', 'Arabic', '    ', true, 15),
('hi', 'Hindi', '   ', false, 16),
('th', 'Thai', '  ', false, 17),
('vi', 'Vietnamese', 'Tiếng Việt', false, 18);

-- ============================================================================
-- 41.2.2 Localization Registry (Master String List)
-- ============================================================================

CREATE TABLE localization_registry (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- String identification
    key VARCHAR(255) NOT NULL UNIQUE,  -- Unique key like 'button.submit', 'error.user_not_
    default_text TEXT NOT NULL,         -- English default text

    -- Categorization
    category VARCHAR(100) NOT NULL,     -- 'ui.buttons', 'errors.validation', 'messages.succ
    subcategory VARCHAR(100),           -- Optional further grouping

    -- Context for translators
    context TEXT,                       -- Description/usage context for translators
    max_length INTEGER,                 -- Character limit if applicable
    placeholders JSONB DEFAULT '[]',    -- List of {name, description} for interpolation

    -- Source tracking
    source_app VARCHAR(50) NOT NULL,    -- 'admin', 'thinktank', 'api', 'shared'
    source_file VARCHAR(255),           -- Original file path where first used

    -- Status
```

```
    is_active BOOLEAN DEFAULT true,
    deprecated_at TIMESTAMPTZ,
    deprecated_replacement_key VARCHAR(255),

    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    created_by UUID REFERENCES administrators(id),

    -- Indexes for fast lookup
    CONSTRAINT valid_key_format CHECK (key ~ '^[a-z][a-z0-9_.]+[a-z0-9]$')
);

CREATE INDEX idx_localization_registry_key ON localization_registry(key);
CREATE INDEX idx_localization_registry_category ON localization_registry(category);
CREATE INDEX idx_localization_registry_source_app ON localization_registry(source_app);
CREATE INDEX idx_localization_registry_active ON localization_registry(is_active) WHERE is_a

-- ============================================================================
-- 41.2.3 Localization Translations (Per-Language Values)
-- ============================================================================

CREATE TYPE translation_status AS ENUM (
    'pending',        -- Not yet translated
    'ai_translated',  -- Auto-translated by AI, needs review
    'in_review',      -- Being reviewed by human
    'approved',       -- Human-approved for production
    'rejected'        -- Rejected, needs re-translation
);

CREATE TABLE localization_translations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Foreign keys
    registry_id UUID NOT NULL REFERENCES localization_registry(id) ON DELETE CASCADE,
    language_code VARCHAR(10) NOT NULL REFERENCES localization_languages(code),

    -- Translation content
    translated_text TEXT NOT NULL,

    -- Status tracking
    status translation_status NOT NULL DEFAULT 'pending',

    -- AI translation metadata
    ai_model VARCHAR(100),           -- e.g., 'anthropic.claude-3-sonnet-20240229-v1:0'
    ai_confidence DECIMAL(3,2),      -- 0.00 to 1.00
```

```
    ai_translated_at TIMESTAMPTZ,

    -- Human review
    reviewed_by UUID REFERENCES administrators(id),
    reviewed_at TIMESTAMPTZ,
    review_notes TEXT,

    -- Version tracking
    version INTEGER NOT NULL DEFAULT 1,
    previous_text TEXT,              -- Previous translation for comparison

    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

    -- Unique constraint
    UNIQUE(registry_id, language_code)
);

CREATE INDEX idx_localization_translations_registry ON localization_translations(registry_id
CREATE INDEX idx_localization_translations_language ON localization_translations(language_co
CREATE INDEX idx_localization_translations_status ON localization_translations(status);
CREATE INDEX idx_localization_translations_needs_review
    ON localization_translations(status)
    WHERE status IN ('ai_translated', 'in_review');

-- ============================================================================
-- 41.2.4 Audit Log for Translation Changes
-- ============================================================================

CREATE TABLE localization_audit_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- What changed
    registry_id UUID REFERENCES localization_registry(id) ON DELETE SET NULL,
    translation_id UUID REFERENCES localization_translations(id) ON DELETE SET NULL,
    language_code VARCHAR(10),

    -- Change details
    action VARCHAR(50) NOT NULL,  -- 'created', 'updated', 'approved', 'rejected', 'ai_trans
    old_value JSONB,
    new_value JSONB,

    -- Who made the change
    changed_by UUID REFERENCES administrators(id),
    changed_by_system BOOLEAN DEFAULT false,  -- True if AI/system change
```

```
    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_localization_audit_registry ON localization_audit_log(registry_id);
CREATE INDEX idx_localization_audit_created ON localization_audit_log(created_at DESC);

-- ============================================================================
-- 41.2.5 Translation Queue (For AI Processing)
-- ============================================================================

CREATE TABLE localization_translation_queue (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    registry_id UUID NOT NULL REFERENCES localization_registry(id) ON DELETE CASCADE,
    language_code VARCHAR(10) NOT NULL REFERENCES localization_languages(code),

    -- Queue status
    status VARCHAR(20) NOT NULL DEFAULT 'pending'
        CHECK (status IN ('pending', 'processing', 'completed', 'failed')),

    -- Processing metadata
    attempts INTEGER DEFAULT 0,
    last_attempt_at TIMESTAMPTZ,
    error_message TEXT,

    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    completed_at TIMESTAMPTZ,

    UNIQUE(registry_id, language_code)
);

CREATE INDEX idx_translation_queue_pending ON localization_translation_queue(status)
    WHERE status = 'pending';

-- ============================================================================
-- 41.2.6 Helper Functions
-- ============================================================================

-- Get translation with fallback chain: requested language -> English -> key
CREATE OR REPLACE FUNCTION get_translation(
    p_key VARCHAR(255),
    p_language VARCHAR(10) DEFAULT 'en'
) RETURNS TEXT AS $$
```

```
DECLARE
    v_result TEXT;
BEGIN
    -- Try requested language first
    SELECT lt.translated_text INTO v_result
    FROM localization_registry lr
    JOIN localization_translations lt ON lt.registry_id = lr.id
    WHERE lr.key = p_key
      AND lt.language_code = p_language
      AND lt.status = 'approved'
      AND lr.is_active = true;

    IF v_result IS NOT NULL THEN
        RETURN v_result;
    END IF;

    -- Fall back to English
    IF p_language != 'en' THEN
        SELECT lt.translated_text INTO v_result
        FROM localization_registry lr
        JOIN localization_translations lt ON lt.registry_id = lr.id
        WHERE lr.key = p_key
          AND lt.language_code = 'en'
          AND lt.status = 'approved'
          AND lr.is_active = true;

        IF v_result IS NOT NULL THEN
            RETURN v_result;
        END IF;
    END IF;

    -- Fall back to default_text from registry
    SELECT lr.default_text INTO v_result
    FROM localization_registry lr
    WHERE lr.key = p_key AND lr.is_active = true;

    -- Return key if nothing found
    RETURN COALESCE(v_result, p_key);
END;
$$ LANGUAGE plpgsql STABLE;

-- Get all translations for a language (for bulk loading)
CREATE OR REPLACE FUNCTION get_all_translations(
    p_language VARCHAR(10) DEFAULT 'en'
) RETURNS TABLE (
    key VARCHAR(255),
```

```
    text TEXT,
    is_fallback BOOLEAN
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        lr.key,
        COALESCE(
            lt_lang.translated_text,
            lt_en.translated_text,
            lr.default_text
        ) as text,
        (lt_lang.translated_text IS NULL) as is_fallback
    FROM localization_registry lr
    LEFT JOIN localization_translations lt_lang
        ON lt_lang.registry_id = lr.id
        AND lt_lang.language_code = p_language
        AND lt_lang.status = 'approved'
    LEFT JOIN localization_translations lt_en
        ON lt_en.registry_id = lr.id
        AND lt_en.language_code = 'en'
        AND lt_en.status = 'approved'
    WHERE lr.is_active = true;
END;
$$ LANGUAGE plpgsql STABLE;

-- Get translation coverage statistics
CREATE OR REPLACE FUNCTION get_translation_coverage()
RETURNS TABLE (
    language_code VARCHAR(10),
    language_name VARCHAR(100),
    total_strings BIGINT,
    translated_count BIGINT,
    approved_count BIGINT,
    ai_translated_count BIGINT,
    pending_count BIGINT,
    coverage_percent DECIMAL(5,2)
) AS $$
BEGIN
    RETURN QUERY
    WITH total AS (
        SELECT COUNT(*) as cnt FROM localization_registry WHERE is_active = true
    )
    SELECT
        ll.code as language_code,
        ll.name as language_name,
```

```
        t.cnt as total_strings,
        COUNT(lt.id) as translated_count,
        COUNT(lt.id) FILTER (WHERE lt.status = 'approved') as approved_count,
        COUNT(lt.id) FILTER (WHERE lt.status = 'ai_translated') as ai_translated_count,
        t.cnt - COUNT(lt.id) as pending_count,
        ROUND((COUNT(lt.id) FILTER (WHERE lt.status = 'approved')::DECIMAL / NULLIF(t.cnt, (
    FROM localization_languages ll
    CROSS JOIN total t
    LEFT JOIN localization_registry lr ON lr.is_active = true
    LEFT JOIN localization_translations lt
        ON lt.registry_id = lr.id
        AND lt.language_code = ll.code
    WHERE ll.is_active = true
    GROUP BY ll.code, ll.name, ll.display_order, t.cnt
    ORDER BY ll.display_order;
END;
$$ LANGUAGE plpgsql STABLE;

-- Trigger to auto-queue translations when new registry entry added
CREATE OR REPLACE FUNCTION queue_translations_for_new_entry()
RETURNS TRIGGER AS $$
BEGIN
    -- Queue translations for all active languages except English
    INSERT INTO localization_translation_queue (registry_id, language_code)
    SELECT NEW.id, ll.code
    FROM localization_languages ll
    WHERE ll.is_active = true AND ll.code != 'en';

    -- Auto-create English translation from default_text
    INSERT INTO localization_translations (registry_id, language_code, translated_text, stat
    VALUES (NEW.id, 'en', NEW.default_text, 'approved');

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_queue_translations
    AFTER INSERT ON localization_registry
    FOR EACH ROW
    EXECUTE FUNCTION queue_translations_for_new_entry();

-- Trigger to log translation changes
CREATE OR REPLACE FUNCTION log_translation_changes()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO localization_audit_log (
```

```
        registry_id,
        translation_id,
        language_code,
        action,
        old_value,
        new_value,
        changed_by,
        changed_by_system
    ) VALUES (
        NEW.registry_id,
        NEW.id,
        NEW.language_code,
        CASE
            WHEN TG_OP = 'INSERT' THEN 'created'
            WHEN OLD.status != NEW.status AND NEW.status = 'approved' THEN 'approved'
            WHEN OLD.status != NEW.status AND NEW.status = 'rejected' THEN 'rejected'
            ELSE 'updated'
        END,
        CASE WHEN TG_OP = 'UPDATE' THEN jsonb_build_object(
            'text', OLD.translated_text,
            'status', OLD.status
        ) END,
        jsonb_build_object(
            'text', NEW.translated_text,
            'status', NEW.status
        ),
        NEW.reviewed_by,
        NEW.ai_model IS NOT NULL AND NEW.reviewed_by IS NULL
    );
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_translation_changes
    AFTER INSERT OR UPDATE ON localization_translations
    FOR EACH ROW
    EXECUTE FUNCTION log_translation_changes();
```

41.3 TYPESCRIPT TYPES & CONSTANTS

File: `packages/shared/src/i18n/types.ts`

```
// ============================================================================
// RADIANT v4.7.0 - Internationalization Types
// ============================================================================
```

```typescript
/**
 * Supported language codes
 */
export const SUPPORTED_LANGUAGES = [
  'en', 'es', 'fr', 'de', 'pt', 'it', 'nl', 'pl', 'ru', 'tr',
  'ja', 'ko', 'zh-CN', 'zh-TW', 'ar', 'hi', 'th', 'vi'
] as const;

export type LanguageCode = typeof SUPPORTED_LANGUAGES[number];

export const DEFAULT_LANGUAGE: LanguageCode = 'en';

export const RTL_LANGUAGES: LanguageCode[] = ['ar'];

/**
 * Language metadata
 */
export interface Language {
  code: LanguageCode;
  name: string;
  nativeName: string;
  isRtl: boolean;
  isActive: boolean;
  displayOrder: number;
}

/**
 * Translation status enum
 */
export type TranslationStatus =
  | 'pending'
  | 'ai_translated'
  | 'in_review'
  | 'approved'
  | 'rejected';

/**
 * Localization registry entry
 */
export interface LocalizationEntry {
  id: string;
  key: string;
  defaultText: string;
  category: string;
  subcategory?: string;
```

```
  context?: string;
  maxLength?: number;
  placeholders?: Array<{
    name: string;
    description: string;
  }>;
  sourceApp: 'admin' | 'thinktank' | 'api' | 'shared';
  sourceFile?: string;
  isActive: boolean;
  deprecatedAt?: string;
  deprecatedReplacementKey?: string;
  createdAt: string;
  updatedAt: string;
}

/**
 * Translation for a specific language
 */
export interface Translation {
  id: string;
  registryId: string;
  languageCode: LanguageCode;
  translatedText: string;
  status: TranslationStatus;
  aiModel?: string;
  aiConfidence?: number;
  aiTranslatedAt?: string;
  reviewedBy?: string;
  reviewedAt?: string;
  reviewNotes?: string;
  version: number;
  previousText?: string;
  createdAt: string;
  updatedAt: string;
}

/**
 * Translation with key (for client-side use)
 */
export interface TranslationBundle {
  [key: string]: string;
}

/**
 * Translation coverage statistics
 */
```

```typescript
export interface TranslationCoverage {
  languageCode: LanguageCode;
  languageName: string;
  totalStrings: number;
  translatedCount: number;
  approvedCount: number;
  aiTranslatedCount: number;
  pendingCount: number;
  coveragePercent: number;
}

/**
 * Interpolation values for placeholders
 */
export type InterpolationValues = Record<string, string | number>;

/**
 * Categories for organizing translations
 */
export const TRANSLATION_CATEGORIES = {
  // UI Elements
  'ui.buttons': 'Buttons and Actions',
  'ui.labels': 'Form Labels',
  'ui.placeholders': 'Input Placeholders',
  'ui.tooltips': 'Tooltips and Hints',
  'ui.navigation': 'Navigation Items',
  'ui.headings': 'Page and Section Headings',

  // Messages
  'messages.success': 'Success Messages',
  'messages.info': 'Informational Messages',
  'messages.warning': 'Warning Messages',
  'messages.loading': 'Loading States',

  // Errors
  'errors.validation': 'Validation Errors',
  'errors.auth': 'Authentication Errors',
  'errors.api': 'API Errors',
  'errors.network': 'Network Errors',
  'errors.system': 'System Errors',

  // Features
  'features.chat': 'Chat Feature',
  'features.thinktank': 'Think Tank Feature',
  'features.admin': 'Admin Dashboard',
  'features.billing': 'Billing & Payments',
```

```
  'features.settings': 'User Settings',

  // Content
  'content.legal': 'Legal Content',
  'content.marketing': 'Marketing Copy',
  'content.help': 'Help & Documentation',
} as const;

export type TranslationCategory = keyof typeof TRANSLATION_CATEGORIES;
```

File: `packages/shared/src/i18n/constants.ts`

```
// ============================================================================
// RADIANT v4.7.0 - i18n Constants
// ============================================================================

import { LanguageCode } from './types';

/**
 * Language metadata mapping
 */
export const LANGUAGE_METADATA: Record<LanguageCode, {
  name: string;
  nativeName: string;
  isRtl: boolean;
}> = {
  'en': { name: 'English', nativeName: 'English', isRtl: false },
  'es': { name: 'Spanish', nativeName: 'Español', isRtl: false },
  'fr': { name: 'French', nativeName: 'Français', isRtl: false },
  'de': { name: 'German', nativeName: 'Deutsch', isRtl: false },
  'pt': { name: 'Portuguese', nativeName: 'Português', isRtl: false },
  'it': { name: 'Italian', nativeName: 'Italiano', isRtl: false },
  'nl': { name: 'Dutch', nativeName: 'Nederlands', isRtl: false },
  'pl': { name: 'Polish', nativeName: 'Polski', isRtl: false },
  'ru': { name: 'Russian', nativeName: '    ', isRtl: false },
  'tr': { name: 'Turkish', nativeName: 'Türkçe', isRtl: false },
  'ja': { name: 'Japanese', nativeName: '  ', isRtl: false },
  'ko': { name: 'Korean', nativeName: '  ', isRtl: false },
  'zh-CN': { name: 'Chinese (Simplified)', nativeName: '  ', isRtl: false },
  'zh-TW': { name: 'Chinese (Traditional)', nativeName: '  ', isRtl: false },
  'ar': { name: 'Arabic', nativeName: '    ', isRtl: true },
  'hi': { name: 'Hindi', nativeName: '   ', isRtl: false },
  'th': { name: 'Thai', nativeName: '  ', isRtl: false },
  'vi': { name: 'Vietnamese', nativeName: 'Tiếng Việt', isRtl: false },
};
```

```typescript
/**
 * Bedrock model for translations
 */
export const TRANSLATION_AI_MODEL = 'anthropic.claude-3-sonnet-20240229-v1:0';

/**
 * Translation system prompt
 */
export const TRANSLATION_SYSTEM_PROMPT = `You are a professional translator for a software a
Your task is to translate UI text, error messages, and user-facing content.

Guidelines:
1. Maintain the same tone and formality level as the source
2. Preserve all placeholders like {name}, {count}, etc. exactly as they appear
3. Keep technical terms consistent with industry standards for the target language
4. For UI elements (buttons, labels), keep translations concise
5. Respect character limits if specified
6. Consider the context provided to ensure accurate translation
7. For RTL languages (Arabic), ensure text flows correctly

Output ONLY the translated text, nothing else.`;

/**
 * Rate limits for translation API
 */
export const TRANSLATION_RATE_LIMITS = {
  maxConcurrent: 10,
  maxPerMinute: 100,
  maxPerHour: 1000,
  retryAttempts: 3,
  retryDelayMs: 1000,
};
```

41.4 AI TRANSLATION LAMBDA

File: `lambda/localization/translate.ts`

```typescript
// ============================================================================
// RADIANT v4.7.0 - AI Translation Lambda
// Triggered by SQS queue when new translation needed
// ============================================================================

import { SQSHandler, SQSEvent } from 'aws-lambda';
import {
  BedrockRuntimeClient,
```

```
  InvokeModelCommand
} from '@aws-sdk/client-bedrock-runtime';
import { Pool } from 'pg';
import { SNSClient, PublishCommand } from '@aws-sdk/client-sns';
import {
  TRANSLATION_AI_MODEL,
  TRANSLATION_SYSTEM_PROMPT,
  LANGUAGE_METADATA
} from '@radiant/shared/i18n';

const bedrock = new BedrockRuntimeClient({ region: process.env.AWS_REGION });
const sns = new SNSClient({ region: process.env.AWS_REGION });
const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false },
});

interface TranslationQueueMessage {
  registryId: string;
  languageCode: string;
  key: string;
  defaultText: string;
  context?: string;
  maxLength?: number;
  placeholders?: Array<{ name: string; description: string }>;
}

export const handler: SQSHandler = async (event: SQSEvent) => {
  for (const record of event.Records) {
    const message: TranslationQueueMessage = JSON.parse(record.body);

    try {
      await translateAndStore(message);
    } catch (error) {
      console.error('Translation failed:', error);
      await updateQueueStatus(message.registryId, message.languageCode, 'failed', error.mess
      throw error; // Re-throw to trigger SQS retry
    }
  }
};

async function translateAndStore(message: TranslationQueueMessage): Promise<void> {
  const { registryId, languageCode, key, defaultText, context, maxLength, placeholders } = m

  // Update queue status to processing
  await updateQueueStatus(registryId, languageCode, 'processing');
```

```
  // Build translation prompt
  const targetLanguage = LANGUAGE_METADATA[languageCode as keyof typeof LANGUAGE_METADATA];

  let userPrompt = `Translate the following English text to ${targetLanguage.name} (${target

Text to translate: "${defaultText}"`;

  if (context) {
    userPrompt += `\n\nContext: ${context}`;
  }

  if (maxLength) {
    userPrompt += `\n\nMaximum length: ${maxLength} characters`;
  }

  if (placeholders && placeholders.length > 0) {
    userPrompt += `\n\nPlaceholders to preserve exactly:`;
    placeholders.forEach(p => {
      userPrompt += `\n- {${p.name}}: ${p.description}`;
    });
  }

  // Call Bedrock
  const response = await bedrock.send(new InvokeModelCommand({
    modelId: TRANSLATION_AI_MODEL,
    contentType: 'application/json',
    accept: 'application/json',
    body: JSON.stringify({
      anthropic_version: 'bedrock-2023-05-31',
      max_tokens: 1024,
      system: TRANSLATION_SYSTEM_PROMPT,
      messages: [
        { role: 'user', content: userPrompt }
      ],
    }),
  }));

  const responseBody = JSON.parse(new TextDecoder().decode(response.body));
  const translatedText = responseBody.content[0].text.trim();

  // Validate placeholders are preserved
  if (placeholders) {
    for (const p of placeholders) {
      if (!translatedText.includes(`{${p.name}}`)) {
        throw new Error(`Translation missing placeholder: {${p.name}}`);
```

```
      }
    }
  }

  // Store translation
  await pool.query(`
    INSERT INTO localization_translations (
      registry_id, language_code, translated_text, status,
      ai_model, ai_confidence, ai_translated_at
    ) VALUES ($1, $2, $3, 'ai_translated', $4, $5, NOW())
    ON CONFLICT (registry_id, language_code)
    DO UPDATE SET
      translated_text = $3,
      status = 'ai_translated',
      ai_model = $4,
      ai_confidence = $5,
      ai_translated_at = NOW(),
      previous_text = localization_translations.translated_text,
      version = localization_translations.version + 1,
      updated_at = NOW()
  `, [registryId, languageCode, translatedText, TRANSLATION_AI_MODEL, 0.85]);

  // Update queue status
  await updateQueueStatus(registryId, languageCode, 'completed');

  // Notify admin of new AI translation needing review
  await notifyAdminOfNewTranslation(key, languageCode, translatedText);
}

async function updateQueueStatus(
  registryId: string,
  languageCode: string,
  status: string,
  errorMessage?: string
): Promise<void> {
  await pool.query(`
    UPDATE localization_translation_queue
    SET status = $3,
        last_attempt_at = NOW(),
        attempts = attempts + 1,
        error_message = $4,
        completed_at = CASE WHEN $3 = 'completed' THEN NOW() ELSE NULL END
    WHERE registry_id = $1 AND language_code = $2
  `, [registryId, languageCode, status, errorMessage]);
}
```

```
async function notifyAdminOfNewTranslation(
  key: string,
  languageCode: string,
  translatedText: string
): Promise<void> {
  const targetLanguage = LANGUAGE_METADATA[languageCode as keyof typeof LANGUAGE_METADATA];

  await sns.send(new PublishCommand({
    TopicArn: process.env.ADMIN_NOTIFICATION_TOPIC_ARN,
    Subject: `[RADIANT] New AI Translation Needs Review`,
    Message: JSON.stringify({
      type: 'translation_review_needed',
      key,
      languageCode,
      languageName: targetLanguage.name,
      translatedText: translatedText.substring(0, 200) + (translatedText.length > 200 ? '...
      dashboardUrl: `${process.env.ADMIN_URL}/localization?filter=ai_translated`,
    }),
    MessageAttributes: {
      notificationType: {
        DataType: 'String',
        StringValue: 'translation_review',
      },
    },
  }));
}
```

File: `lambda/localization/process-queue.ts`

```
// ============================================================================
// RADIANT v4.7.0 - Translation Queue Processor
// Scheduled Lambda to process pending translations
// ============================================================================

import { ScheduledHandler } from 'aws-lambda';
import { SQSClient, SendMessageCommand } from '@aws-sdk/client-sqs';
import { Pool } from 'pg';

const sqs = new SQSClient({ region: process.env.AWS_REGION });
const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false },
});

export const handler: ScheduledHandler = async () => {
  // Get pending translations (limit batch size)
```

```
  const result = await pool.query(`
    SELECT
      q.registry_id,
      q.language_code,
      r.key,
      r.default_text,
      r.context,
      r.max_length,
      r.placeholders
    FROM localization_translation_queue q
    JOIN localization_registry r ON r.id = q.registry_id
    WHERE q.status = 'pending'
      AND q.attempts < 3
      AND r.is_active = true
    ORDER BY q.created_at ASC
    LIMIT 50
  `);

  console.log(`Processing ${result.rows.length} pending translations`);

  for (const row of result.rows) {
    await sqs.send(new SendMessageCommand({
      QueueUrl: process.env.TRANSLATION_QUEUE_URL,
      MessageBody: JSON.stringify({
        registryId: row.registry_id,
        languageCode: row.language_code,
        key: row.key,
        defaultText: row.default_text,
        context: row.context,
        maxLength: row.max_length,
        placeholders: row.placeholders,
      }),
      MessageGroupId: row.language_code, // FIFO queue grouping
      MessageDeduplicationId: `${row.registry_id}-${row.language_code}-${Date.now()}`,
    }));
  }

  return {
    processed: result.rows.length,
  };
};
```

41.5 LOCALIZATION SERVICE (API)

File: `lambda/localization/api.ts`

```
// ============================================================================
// RADIANT v4.7.0 - Localization API Lambda
// ============================================================================

import { APIGatewayProxyHandler, APIGatewayProxyResult } from 'aws-lambda';
import { Pool } from 'pg';
import { extractAuthContext, requireRoles } from '../shared/auth';
import {
  LocalizationEntry,
  Translation,
  TranslationCoverage,
  LanguageCode,
  SUPPORTED_LANGUAGES
} from '@radiant/shared/i18n';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false },
});

export const handler: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxyResult>
  const { httpMethod, path, pathParameters, queryStringParameters, body } = event;

  try {
    // Public endpoint: get translations bundle
    if (httpMethod === 'GET' && path.match(/\/localization\/bundle\/[a-z-]+$/)) {
      return await getTranslationBundle(pathParameters?.language as LanguageCode);
    }

    // All other endpoints require authentication
    const auth = extractAuthContext(event);

    // GET /localization/languages - Get supported languages
    if (httpMethod === 'GET' && path === '/localization/languages') {
      return await getLanguages();
    }

    // GET /localization/coverage - Get translation coverage stats
    if (httpMethod === 'GET' && path === '/localization/coverage') {
      requireRoles(auth, ['admin', 'super_admin']);
      return await getCoverage();
    }
```

```
// GET /localization/registry - List all registry entries
if (httpMethod === 'GET' && path === '/localization/registry') {
  requireRoles(auth, ['admin', 'super_admin']);
  return await listRegistry(queryStringParameters);
}

// POST /localization/registry - Create new registry entry
if (httpMethod === 'POST' && path === '/localization/registry') {
  requireRoles(auth, ['admin', 'super_admin']);
  return await createRegistryEntry(JSON.parse(body || '{}'), auth.userId);
}

// PUT /localization/registry/:id - Update registry entry
if (httpMethod === 'PUT' && path.match(/\/localization\/registry\/[a-f0-9-]+$/)) {
  requireRoles(auth, ['admin', 'super_admin']);
  return await updateRegistryEntry(pathParameters?.id!, JSON.parse(body || '{}'));
}

// GET /localization/translations/:registryId - Get translations for entry
if (httpMethod === 'GET' && path.match(/\/localization\/translations\/[a-f0-9-]+$/)) {
  requireRoles(auth, ['admin', 'super_admin']);
  return await getTranslations(pathParameters?.registryId!);
}

// PUT /localization/translations/:id - Update translation
if (httpMethod === 'PUT' && path.match(/\/localization\/translations\/[a-f0-9-]+$/)) {
  requireRoles(auth, ['admin', 'super_admin']);
  return await updateTranslation(pathParameters?.id!, JSON.parse(body || '{}'), auth.use
}

// POST /localization/translations/:id/approve - Approve translation
if (httpMethod === 'POST' && path.match(/\/localization\/translations\/[a-f0-9-]+\/appro
  requireRoles(auth, ['admin', 'super_admin']);
  return await approveTranslation(pathParameters?.id!, auth.userId);
}

// POST /localization/translations/:id/reject - Reject translation
if (httpMethod === 'POST' && path.match(/\/localization\/translations\/[a-f0-9-]+\/rejec
  requireRoles(auth, ['admin', 'super_admin']);
  return await rejectTranslation(pathParameters?.id!, JSON.parse(body || '{}'), auth.use
}

// GET /localization/pending - Get translations needing review
if (httpMethod === 'GET' && path === '/localization/pending') {
  requireRoles(auth, ['admin', 'super_admin']);
```

```
      return await getPendingReviews(queryStringParameters);
    }

    // POST /localization/bulk-approve - Bulk approve translations
    if (httpMethod === 'POST' && path === '/localization/bulk-approve') {
      requireRoles(auth, ['super_admin']);
      return await bulkApprove(JSON.parse(body || '{}'), auth.userId);
    }

    return { statusCode: 404, body: JSON.stringify({ error: 'Not found' }) };
  } catch (error) {
    console.error('Localization API error:', error);
    return {
      statusCode: error.statusCode || 500,
      body: JSON.stringify({ error: error.message }),
    };
  }
};

// Get translation bundle for client
async function getTranslationBundle(language: LanguageCode): Promise<APIGatewayProxyResult>
  if (!SUPPORTED_LANGUAGES.includes(language)) {
    return { statusCode: 400, body: JSON.stringify({ error: 'Unsupported language' }) };
  }

  const result = await pool.query(`SELECT * FROM get_all_translations($1)`, [language]);

  const bundle: Record<string, string> = {};
  for (const row of result.rows) {
    bundle[row.key] = row.text;
  }

  return {
    statusCode: 200,
    headers: {
      'Content-Type': 'application/json',
      'Cache-Control': 'public, max-age=300', // Cache for 5 minutes
    },
    body: JSON.stringify(bundle),
  };
}

async function getLanguages(): Promise<APIGatewayProxyResult> {
  const result = await pool.query(`
    SELECT code, name, native_name, is_rtl, is_active, display_order
    FROM localization_languages
```

```
    WHERE is_active = true
    ORDER BY display_order
  `);

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows.map(row => ({
      code: row.code,
      name: row.name,
      nativeName: row.native_name,
      isRtl: row.is_rtl,
      displayOrder: row.display_order,
    }))),
  };
}

async function getCoverage(): Promise<APIGatewayProxyResult> {
  const result = await pool.query(`SELECT * FROM get_translation_coverage()`);

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows.map(row => ({
      languageCode: row.language_code,
      languageName: row.language_name,
      totalStrings: parseInt(row.total_strings),
      translatedCount: parseInt(row.translated_count),
      approvedCount: parseInt(row.approved_count),
      aiTranslatedCount: parseInt(row.ai_translated_count),
      pendingCount: parseInt(row.pending_count),
      coveragePercent: parseFloat(row.coverage_percent),
    }))),
  };
}

async function listRegistry(params: any): Promise<APIGatewayProxyResult> {
  const { category, sourceApp, search, page = '1', limit = '50' } = params || {};
  const offset = (parseInt(page) - 1) * parseInt(limit);

  let query = `
    SELECT lr.*,
           (SELECT COUNT(*) FROM localization_translations lt WHERE lt.registry_id = lr.id A
           (SELECT COUNT(*) FROM localization_translations lt WHERE lt.registry_id = lr.id A
    FROM localization_registry lr
    WHERE lr.is_active = true
  `;
  const queryParams: any[] = [];
```

```
  let paramIndex = 1;

  if (category) {
    query += ` AND lr.category = $${paramIndex++}`;
    queryParams.push(category);
  }

  if (sourceApp) {
    query += ` AND lr.source_app = $${paramIndex++}`;
    queryParams.push(sourceApp);
  }

  if (search) {
    query += ` AND (lr.key ILIKE $${paramIndex} OR lr.default_text ILIKE $${paramIndex})`;
    queryParams.push(`%${search}%`);
    paramIndex++;
  }

  query += ` ORDER BY lr.category, lr.key LIMIT $${paramIndex++} OFFSET $${paramIndex}`;
  queryParams.push(parseInt(limit), offset);

  const result = await pool.query(query, queryParams);

  // Get total count
  const countResult = await pool.query(`
    SELECT COUNT(*) FROM localization_registry WHERE is_active = true
  `);

  return {
    statusCode: 200,
    body: JSON.stringify({
      data: result.rows,
      total: parseInt(countResult.rows[0].count),
      page: parseInt(page),
      limit: parseInt(limit),
    }),
  };
}

async function createRegistryEntry(data: any, userId: string): Promise<APIGatewayProxyResult
  const { key, defaultText, category, subcategory, context, maxLength, placeholders, sourceA

  // Validate key format
  if (!/^[a-z][a-z0-9_.]+[a-z0-9]$/.test(key)) {
    return {
      statusCode: 400,
```

```
      body: JSON.stringify({ error: 'Invalid key format. Use lowercase with dots/underscores
    };
  }

  const result = await pool.query(`
    INSERT INTO localization_registry (
      key, default_text, category, subcategory, context,
      max_length, placeholders, source_app, source_file, created_by
    ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10)
    RETURNING *
  `, [key, defaultText, category, subcategory, context, maxLength,
      JSON.stringify(placeholders || []), sourceApp, sourceFile, userId]);

  return {
    statusCode: 201,
    body: JSON.stringify(result.rows[0]),
  };
}

async function updateTranslation(id: string, data: any, userId: string): Promise<APIGatewayP
  const { translatedText, status } = data;

  const result = await pool.query(`
    UPDATE localization_translations
    SET translated_text = COALESCE($2, translated_text),
        status = COALESCE($3, status),
        reviewed_by = $4,
        reviewed_at = NOW(),
        previous_text = translated_text,
        version = version + 1,
        updated_at = NOW()
    WHERE id = $1
    RETURNING *
  `, [id, translatedText, status, userId]);

  if (result.rows.length === 0) {
    return { statusCode: 404, body: JSON.stringify({ error: 'Translation not found' }) };
  }

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows[0]),
  };
}

async function approveTranslation(id: string, userId: string): Promise<APIGatewayProxyResult
```

```
  const result = await pool.query(`
    UPDATE localization_translations
    SET status = 'approved',
        reviewed_by = $2,
        reviewed_at = NOW(),
        updated_at = NOW()
    WHERE id = $1
    RETURNING *
  `, [id, userId]);

  if (result.rows.length === 0) {
    return { statusCode: 404, body: JSON.stringify({ error: 'Translation not found' }) };
  }

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows[0]),
  };
}

async function rejectTranslation(id: string, data: any, userId: string): Promise<APIGatewayP
  const { reviewNotes } = data;

  const result = await pool.query(`
    UPDATE localization_translations
    SET status = 'rejected',
        reviewed_by = $2,
        reviewed_at = NOW(),
        review_notes = $3,
        updated_at = NOW()
    WHERE id = $1
    RETURNING *
  `, [id, userId, reviewNotes]);

  // Re-queue for translation
  await pool.query(`
    INSERT INTO localization_translation_queue (registry_id, language_code)
    SELECT registry_id, language_code FROM localization_translations WHERE id = $1
    ON CONFLICT (registry_id, language_code) DO UPDATE SET
      status = 'pending',
      attempts = 0,
      error_message = NULL
  `, [id]);

  return {
    statusCode: 200,
```

```
    body: JSON.stringify(result.rows[0]),
  };
}

async function getPendingReviews(params: any): Promise<APIGatewayProxyResult> {
  const { language, page = '1', limit = '50' } = params || {};
  const offset = (parseInt(page) - 1) * parseInt(limit);

  let query = `
    SELECT lt.*, lr.key, lr.default_text, lr.context, lr.category,
           ll.name as language_name, ll.native_name
    FROM localization_translations lt
    JOIN localization_registry lr ON lr.id = lt.registry_id
    JOIN localization_languages ll ON ll.code = lt.language_code
    WHERE lt.status = 'ai_translated'
  `;
  const queryParams: any[] = [];
  let paramIndex = 1;

  if (language) {
    query += ` AND lt.language_code = $${paramIndex++}`;
    queryParams.push(language);
  }

  query += ` ORDER BY lt.ai_translated_at DESC LIMIT $${paramIndex++} OFFSET $${paramIndex}`
  queryParams.push(parseInt(limit), offset);

  const result = await pool.query(query, queryParams);

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows),
  };
}

async function bulkApprove(data: any, userId: string): Promise<APIGatewayProxyResult> {
  const { translationIds } = data;

  if (!Array.isArray(translationIds) || translationIds.length === 0) {
    return { statusCode: 400, body: JSON.stringify({ error: 'translationIds required' }) };
  }

  const result = await pool.query(`
    UPDATE localization_translations
    SET status = 'approved',
        reviewed_by = $2,
```

```
        reviewed_at = NOW(),
        updated_at = NOW()
    WHERE id = ANY($1::uuid[])
    RETURNING id
  `, [translationIds, userId]);

  return {
    statusCode: 200,
    body: JSON.stringify({ approved: result.rows.length }),
  };
}

// Export individual functions for updateRegistryEntry and getTranslations
async function updateRegistryEntry(id: string, data: any): Promise<APIGatewayProxyResult> {
  const { defaultText, category, subcategory, context, maxLength, placeholders, isActive } =

  const result = await pool.query(`
    UPDATE localization_registry
    SET default_text = COALESCE($2, default_text),
        category = COALESCE($3, category),
        subcategory = COALESCE($4, subcategory),
        context = COALESCE($5, context),
        max_length = COALESCE($6, max_length),
        placeholders = COALESCE($7, placeholders),
        is_active = COALESCE($8, is_active),
        updated_at = NOW()
    WHERE id = $1
    RETURNING *
  `, [id, defaultText, category, subcategory, context, maxLength,
      placeholders ? JSON.stringify(placeholders) : null, isActive]);

  if (result.rows.length === 0) {
    return { statusCode: 404, body: JSON.stringify({ error: 'Registry entry not found' }) };
  }

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows[0]),
  };
}

async function getTranslations(registryId: string): Promise<APIGatewayProxyResult> {
  const result = await pool.query(`
    SELECT lt.*, ll.name as language_name, ll.native_name, ll.is_rtl
    FROM localization_translations lt
    JOIN localization_languages ll ON ll.code = lt.language_code
```

```
    WHERE lt.registry_id = $1
    ORDER BY ll.display_order
  `, [registryId]);

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows),
  };
}
```

41.6 REACT i18n IMPLEMENTATION

File: `packages/shared/src/i18n/react/I18nProvider.tsx`

```
// ═══════════════════════════════════════════════════════════════════════════
// RADIANT v4.7.0 - React i18n Provider
// ═══════════════════════════════════════════════════════════════════════════

import React, {
  createContext,
  useContext,
  useState,
  useEffect,
  useCallback,
  ReactNode
} from 'react';
import {
  LanguageCode,
  TranslationBundle,
  InterpolationValues,
  DEFAULT_LANGUAGE,
  RTL_LANGUAGES,
  LANGUAGE_METADATA
} from '../types';

interface I18nContextType {
  language: LanguageCode;
  setLanguage: (lang: LanguageCode) => void;
  t: (key: string, values?: InterpolationValues) => string;
  isRtl: boolean;
  isLoading: boolean;
  languages: typeof LANGUAGE_METADATA;
}

const I18nContext = createContext<I18nContextType | null>(null);
```

```typescript
interface I18nProviderProps {
  children: ReactNode;
  defaultLanguage?: LanguageCode;
  apiBaseUrl: string;
}

export function I18nProvider({
  children,
  defaultLanguage = DEFAULT_LANGUAGE,
  apiBaseUrl
}: I18nProviderProps) {
  const [language, setLanguageState] = useState<LanguageCode>(() => {
    // Try to get from localStorage first
    if (typeof window !== 'undefined') {
      const stored = localStorage.getItem('radiant_language');
      if (stored && Object.keys(LANGUAGE_METADATA).includes(stored)) {
        return stored as LanguageCode;
      }
    }
    return defaultLanguage;
  });

  const [translations, setTranslations] = useState<TranslationBundle>({});
  const [isLoading, setIsLoading] = useState(true);

  // Load translations when language changes
  useEffect(() => {
    let cancelled = false;

    async function loadTranslations() {
      setIsLoading(true);
      try {
        const response = await fetch(`${apiBaseUrl}/localization/bundle/${language}`);
        if (!response.ok) throw new Error('Failed to load translations');
        const bundle = await response.json();
        if (!cancelled) {
          setTranslations(bundle);
        }
      } catch (error) {
        console.error('Failed to load translations:', error);
        // Keep existing translations on error
      } finally {
        if (!cancelled) {
          setIsLoading(false);
        }
```

```
    }
  }

  loadTranslations();

  return () => {
    cancelled = true;
  };
}, [language, apiBaseUrl]);

// Update document direction for RTL languages
useEffect(() => {
  if (typeof document !== 'undefined') {
    document.documentElement.dir = RTL_LANGUAGES.includes(language) ? 'rtl' : 'ltr';
    document.documentElement.lang = language;
  }
}, [language]);

const setLanguage = useCallback((lang: LanguageCode) => {
  setLanguageState(lang);
  if (typeof window !== 'undefined') {
    localStorage.setItem('radiant_language', lang);
  }
}, []);

// Translation function with interpolation
const t = useCallback((key: string, values?: InterpolationValues): string => {
  let text = translations[key] || key;

  // Interpolate values
  if (values) {
    Object.entries(values).forEach(([k, v]) => {
      text = text.replace(new RegExp(`\\{${k}\\}`, 'g'), String(v));
    });
  }

  return text;
}, [translations]);

const value: I18nContextType = {
  language,
  setLanguage,
  t,
  isRtl: RTL_LANGUAGES.includes(language),
  isLoading,
  languages: LANGUAGE_METADATA,
```

```
  };

  return (
    <I18nContext.Provider value={value}>
      {children}
    </I18nContext.Provider>
  );
}

/**
 * Hook to access i18n context
 */
export function useI18n(): I18nContextType {
  const context = useContext(I18nContext);
  if (!context) {
    throw new Error('useI18n must be used within an I18nProvider');
  }
  return context;
}

/**
 * Hook for translation function only
 */
export function useTranslation() {
  const { t, language, isRtl } = useI18n();
  return { t, language, isRtl };
}
```

File: `packages/shared/src/i18n/react/LanguageSelector.tsx`

```
// ============================================================================
// RADIANT v4.7.0 - Language Selector Component
// ============================================================================

import React from 'react';
import { useI18n } from './I18nProvider';
import { LanguageCode } from '../types';

interface LanguageSelectorProps {
  className?: string;
  showNativeName?: boolean;
  compact?: boolean;
}

export function LanguageSelector({
  className = '',
```

```
  showNativeName = true,
  compact = false
}: LanguageSelectorProps) {
  const { language, setLanguage, languages } = useI18n();

  const handleChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
    setLanguage(e.target.value as LanguageCode);
  };

  return (
    <select
      value={language}
      onChange={handleChange}
      className={`language-selector ${className}`}
      aria-label="Select language"
    >
      {Object.entries(languages).map(([code, meta]) => (
        <option key={code} value={code}>
          {compact
            ? code.toUpperCase()
            : showNativeName
              ? `${meta.nativeName} (${meta.name})`
              : meta.name
          }
        </option>
      ))}
    </select>
  );
}
```

File: `packages/shared/src/i18n/react/Trans.tsx`

```
// ============================================================================
// RADIANT v4.7.0 - Trans Component for complex translations
// ============================================================================

import React, { ReactNode } from 'react';
import { useTranslation } from './I18nProvider';
import { InterpolationValues } from '../types';

interface TransProps {
  i18nKey: string;
  values?: InterpolationValues;
  components?: Record<string, ReactNode>;
  fallback?: string;
}
```

```
/**
 * Trans component for translations with embedded React components
 *
 * Usage:
 * <Trans
 *   i18nKey="message.welcome"
 *   values={{ name: 'John' }}
 *   components={{ bold: <strong />, link: <a href="/profile" /> }}
 * />
 *
 * Translation: "Hello <bold>{name}</bold>! Visit your <link>profile</link>."
 */
export function Trans({ i18nKey, values, components, fallback }: TransProps) {
  const { t } = useTranslation();

  let text = t(i18nKey);

  // If key not found, use fallback
  if (text === i18nKey && fallback) {
    text = fallback;
  }

  // Interpolate values first
  if (values) {
    Object.entries(values).forEach(([k, v]) => {
      text = text.replace(new RegExp(`\\{${k}\\}`, 'g'), String(v));
    });
  }

  // If no components, return plain text
  if (!components) {
    return <>{text}</>;
  }

  // Parse and replace component placeholders
  const parts: ReactNode[] = [];
  let remaining = text;
  let key = 0;

  Object.entries(components).forEach(([name, component]) => {
    const regex = new RegExp(`<${name}>(.*?)</${name}>`, 'g');
    const newParts: ReactNode[] = [];
    let lastIndex = 0;
    let match;
```

```
    while ((match = regex.exec(remaining)) !== null) {
      // Add text before the match
      if (match.index > lastIndex) {
        newParts.push(remaining.slice(lastIndex, match.index));
      }

      // Clone component with content
      if (React.isValidElement(component)) {
        newParts.push(
          React.cloneElement(component, { key: key++ }, match[1])
        );
      }

      lastIndex = regex.lastIndex;
    }

    // Add remaining text
    if (lastIndex < remaining.length) {
      newParts.push(remaining.slice(lastIndex));
    }

    remaining = newParts.join('');
  });

  return <>{remaining}</>;
}
```

41.7 ESLINT PLUGIN (HARDCODE PREVENTION)

File: `packages/eslint-plugin-i18n/src/index.ts`

```
// ============================================================================
// RADIANT v4.7.0 - ESLint Plugin to Prevent Hardcoded Strings
// ============================================================================

import { ESLintUtils, TSESTree } from '@typescript-eslint/utils';

const createRule = ESLintUtils.RuleCreator(
  (name) => `https://radiant.dev/eslint/${name}`
);

// Patterns that are allowed without translation
const ALLOWED_PATTERNS = [
  /^[a-z][a-z0-9_.]+[a-z0-9]$/,  // Translation keys like 'button.submit'
  /^https?:\/\//,                  // URLs
```

```
  /^[A-Z_]+$/,                    // Constants like 'GET', 'POST'
  /^\d+$/,                        // Pure numbers
  /^[a-z]+\.[a-z]+$/,             // File extensions like 'image.png'
  /^#[0-9a-fA-F]+$/,              // Hex colors
  /^rgb/,                         // RGB colors
  /^data:/,                       // Data URLs
  /^[a-z-]+\/[a-z-]+$/,           // MIME types
  /^\s*$/,                        // Whitespace only
];

// JSX attributes that commonly have hardcoded strings
const ALLOWED_JSX_ATTRIBUTES = [
  'className', 'class', 'id', 'name', 'type', 'href', 'src', 'alt',
  'placeholder', 'title', 'aria-label', 'data-testid', 'key', 'role',
  'target', 'rel', 'method', 'action', 'encType', 'accept', 'pattern',
];

// Function names that accept translation keys
const TRANSLATION_FUNCTIONS = ['t', 'i18n', 'translate', 'L10n'];

export const noHardcodedStrings = createRule({
  name: 'no-hardcoded-strings',
  meta: {
    type: 'suggestion',
    docs: {
      description: 'Disallow hardcoded user-facing strings. Use translation keys instead.',
    },
    messages: {
      hardcodedString:
        'Hardcoded string "{{text}}" detected. Use translation: t(\'{{suggestedKey}}\')',
      hardcodedJsxText:
        'Hardcoded JSX text "{{text}}" detected. Use: {t(\'{{suggestedKey}}\')}',
    },
    schema: [
      {
        type: 'object',
        properties: {
          ignorePaths: {
            type: 'array',
            items: { type: 'string' },
          },
          ignorePatterns: {
            type: 'array',
            items: { type: 'string' },
          },
        },
```

```
    },
  ],
},
defaultOptions: [{ ignorePaths: [], ignorePatterns: [] }],
create(context, [options]) {
  const filename = context.filename || context.getFilename();

  // Skip test files and config files
  if (
    filename.includes('.test.') ||
    filename.includes('.spec.') ||
    filename.includes('.config.') ||
    filename.includes('__tests__') ||
    filename.includes('__mocks__')
  ) {
    return {};
  }

  // Skip files in ignore paths
  if (options.ignorePaths?.some(path => filename.includes(path))) {
    return {};
  }

  function isAllowedString(value: string): boolean {
    // Check built-in patterns
    if (ALLOWED_PATTERNS.some(pattern => pattern.test(value))) {
      return true;
    }

    // Check custom ignore patterns
    if (options.ignorePatterns?.some(pattern => new RegExp(pattern).test(value))) {
      return true;
    }

    // Allow very short strings (likely not user-facing)
    if (value.length <= 2) {
      return true;
    }

    return false;
  }

  function generateSuggestedKey(text: string): string {
    // Generate a key based on the text
    const words = text.toLowerCase()
      .replace(/[^a-z0-9\s]/g, '')
```

```
    .split(/\s+/)
    .slice(0, 4)
    .join('_');
  return `ui.text.${words || 'untitled'}`;
}

function isInsideTranslationCall(node: TSESTree.Node): boolean {
  let parent = node.parent;
  while (parent) {
    if (
      parent.type === 'CallExpression' &&
      parent.callee.type === 'Identifier' &&
      TRANSLATION_FUNCTIONS.includes(parent.callee.name)
    ) {
      return true;
    }
    parent = parent.parent;
  }
  return false;
}

return {
  // Check JSX text content
  JSXText(node) {
    const text = node.value.trim();
    if (text && !isAllowedString(text)) {
      context.report({
        node,
        messageId: 'hardcodedJsxText',
        data: {
          text: text.substring(0, 30) + (text.length > 30 ? '...' : ''),
          suggestedKey: generateSuggestedKey(text),
        },
      });
    }
  },

  // Check string literals in JSX expressions
  'JSXExpressionContainer > Literal'(node: TSESTree.Literal) {
    if (typeof node.value !== 'string') return;
    const text = node.value.trim();

    if (text && !isAllowedString(text) && !isInsideTranslationCall(node)) {
      context.report({
        node,
        messageId: 'hardcodedString',
```

```
      data: {
        text: text.substring(0, 30) + (text.length > 30 ? '...' : ''),
        suggestedKey: generateSuggestedKey(text),
      },
    });
  }
},

// Check JSX attribute values (only specific attributes)
'JSXAttribute > Literal'(node: TSESTree.Literal) {
  if (typeof node.value !== 'string') return;

  const parent = node.parent as TSESTree.JSXAttribute;
  const attrName = parent.name.type === 'JSXIdentifier'
    ? parent.name.name
    : '';

  // Skip allowed attributes
  if (ALLOWED_JSX_ATTRIBUTES.includes(attrName)) {
    return;
  }

  // Check attributes that should be translated
  const translatableAttrs = ['label', 'title', 'placeholder', 'aria-label', 'errorMess
  if (translatableAttrs.includes(attrName)) {
    const text = node.value.trim();
    if (text && !isAllowedString(text)) {
      context.report({
        node,
        messageId: 'hardcodedString',
        data: {
          text: text.substring(0, 30) + (text.length > 30 ? '...' : ''),
          suggestedKey: generateSuggestedKey(text),
        },
      });
    }
  }
},

// Check error messages in throw statements
'ThrowStatement CallExpression'(node: TSESTree.CallExpression) {
  if (node.arguments.length === 0) return;

  const firstArg = node.arguments[0];
  if (firstArg.type === 'Literal' && typeof firstArg.value === 'string') {
    const text = firstArg.value.trim();
```

```
          if (text && !isAllowedString(text) && !isInsideTranslationCall(firstArg)) {
            context.report({
              node: firstArg,
              messageId: 'hardcodedString',
              data: {
                text: text.substring(0, 30) + (text.length > 30 ? '...' : ''),
                suggestedKey: `errors.${generateSuggestedKey(text).replace('ui.text.', '')}`
              },
            });
          }
        }
      },

      // Check object properties that are likely user-facing
      'Property > Literal'(node: TSESTree.Literal) {
        if (typeof node.value !== 'string') return;

        const parent = node.parent as TSESTree.Property;
        if (parent.key.type !== 'Identifier') return;

        const propName = parent.key.name;
        const userFacingProps = ['message', 'title', 'description', 'label', 'text', 'errorM

        if (userFacingProps.includes(propName)) {
          const text = node.value.trim();
          if (text && !isAllowedString(text) && !isInsideTranslationCall(node)) {
            context.report({
              node,
              messageId: 'hardcodedString',
              data: {
                text: text.substring(0, 30) + (text.length > 30 ? '...' : ''),
                suggestedKey: generateSuggestedKey(text),
              },
            });
          }
        }
      },
    };
  },
});

export const rules = {
  'no-hardcoded-strings': noHardcodedStrings,
};

export const configs = {
```

```
  recommended: {
    plugins: ['@radiant/i18n'],
    rules: {
      '@radiant/i18n/no-hardcoded-strings': 'error',
    },
  },
};
```

File: `.eslintrc.js` (Project Root Addition)

```
// Add to existing ESLint config
module.exports = {
  // ... existing config
  plugins: [
    // ... existing plugins
    '@radiant/i18n',
  ],
  rules: {
    // ... existing rules
    '@radiant/i18n/no-hardcoded-strings': ['error', {
      ignorePaths: [
        'node_modules',
        '__tests__',
        '*.test.ts',
        '*.spec.ts',
        'migrations',
      ],
      ignorePatterns: [
        '^[A-Z][A-Z_]+$',  // Constants
        '^/api/',          // API paths
      ],
    }],
  },
};
```

41.8 SWIFT LOCALIZATION SERVICE (THINK TANK APP)

File: `ThinkTank/Services/LocalizationService.swift`

```
// ============================================================================
// RADIANT v4.7.0 - Swift Localization Service
// ============================================================================

import Foundation
```

```
import Combine

/// Supported languages
enum SupportedLanguage: String, CaseIterable, Codable {
    case en = "en"
    case es = "es"
    case fr = "fr"
    case de = "de"
    case pt = "pt"
    case it = "it"
    case nl = "nl"
    case pl = "pl"
    case ru = "ru"
    case tr = "tr"
    case ja = "ja"
    case ko = "ko"
    case zhCN = "zh-CN"
    case zhTW = "zh-TW"
    case ar = "ar"
    case hi = "hi"
    case th = "th"
    case vi = "vi"

    var displayName: String {
        switch self {
        case .en: return "English"
        case .es: return "Español"
        case .fr: return "Français"
        case .de: return "Deutsch"
        case .pt: return "Português"
        case .it: return "Italiano"
        case .nl: return "Nederlands"
        case .pl: return "Polski"
        case .ru: return "    "
        case .tr: return "Türkçe"
        case .ja: return "  "
        case .ko: return "  "
        case .zhCN: return "   "
        case .zhTW: return "   "
        case .ar: return "    "
        case .hi: return "   "
        case .th: return "  "
        case .vi: return "Tiếng Việt"
        }
    }
```

```swift
    var isRTL: Bool {
        self == .ar
    }

    /// Get system preferred language or default to English
    static var preferred: SupportedLanguage {
        let preferredIdentifier = Locale.preferredLanguages.first ?? "en"

        // Try exact match first
        if let exact = SupportedLanguage(rawValue: preferredIdentifier) {
            return exact
        }

        // Try language code only (e.g., "en-US" -> "en")
        let languageCode = String(preferredIdentifier.prefix(2))
        if let lang = SupportedLanguage(rawValue: languageCode) {
            return lang
        }

        // Handle Chinese variants
        if preferredIdentifier.hasPrefix("zh") {
            if preferredIdentifier.contains("Hans") || preferredIdentifier.contains("CN") {
                return .zhCN
            } else {
                return .zhTW
            }
        }

        return .en
    }
}

/// Localization service singleton
@MainActor
final class LocalizationService: ObservableObject {
    static let shared = LocalizationService()

    @Published private(set) var currentLanguage: SupportedLanguage
    @Published private(set) var isLoading = false
    @Published private(set) var translations: [String: String] = [:]

    private let apiBaseURL: URL
    private var cancellables = Set<AnyCancellable>()

    private init() {
        // Load saved language or use system preferred
```

```
        if let savedLang = UserDefaults.standard.string(forKey: "radiant_language"),
           let lang = SupportedLanguage(rawValue: savedLang) {
            self.currentLanguage = lang
        } else {
            self.currentLanguage = .preferred
        }

        self.apiBaseURL = URL(string: Configuration.shared.apiBaseURL)!

        // Load translations for current language
        Task {
            await loadTranslations()
        }
    }

    /// Change current language
    func setLanguage(_ language: SupportedLanguage) async {
        guard language != currentLanguage else { return }

        currentLanguage = language
        UserDefaults.standard.set(language.rawValue, forKey: "radiant_language")

        await loadTranslations()
    }

    /// Load translations from API
    func loadTranslations() async {
        isLoading = true
        defer { isLoading = false }

        do {
            let url = apiBaseURL.appendingPathComponent("localization/bundle/\(currentLangua
            let (data, response) = try await URLSession.shared.data(from: url)

            guard let httpResponse = response as? HTTPURLResponse,
                  httpResponse.statusCode == 200 else {
                throw LocalizationError.networkError
            }

            let bundle = try JSONDecoder().decode([String: String].self, from: data)
            translations = bundle

            // Cache translations locally
            cacheTranslations(bundle)

        } catch {
```

```
                print("Failed to load translations: \(error)")
                // Load from cache on error
                loadCachedTranslations()
            }
        }

        /// Get translated string for key
        func translate(_ key: String, values: [String: Any] = [:]) -> String {
            var text = translations[key] ?? key

            // Interpolate values
            for (name, value) in values {
                text = text.replacingOccurrences(of: "{\(name)}", with: String(describing: value
            }

            return text
        }

        /// Shorthand translation function
        func t(_ key: String, _ values: [String: Any] = [:]) -> String {
            translate(key, values: values)
        }

        // MARK: - Caching

        private func cacheTranslations(_ bundle: [String: String]) {
            guard let data = try? JSONEncoder().encode(bundle) else { return }

            let cacheURL = getCacheURL()
            try? data.write(to: cacheURL)
        }

        private func loadCachedTranslations() {
            let cacheURL = getCacheURL()
            guard let data = try? Data(contentsOf: cacheURL),
                  let bundle = try? JSONDecoder().decode([String: String].self, from: data) else
                return
            }
            translations = bundle
        }

        private func getCacheURL() -> URL {
            let cacheDir = FileManager.default.urls(for: .cachesDirectory, in: .userDomainMask)
            return cacheDir.appendingPathComponent("translations_\(currentLanguage.rawValue).jso
        }
    }
```

```swift
enum LocalizationError: Error {
    case networkError
    case decodingError
}

// MARK: - Property Wrapper for SwiftUI

@propertyWrapper
struct Localized: DynamicProperty {
    @ObservedObject private var service = LocalizationService.shared
    private let key: String
    private let values: [String: Any]

    init(_ key: String, values: [String: Any] = [:]) {
        self.key = key
        self.values = values
    }

    var wrappedValue: String {
        service.translate(key, values: values)
    }
}

// MARK: - SwiftUI Extensions

extension View {
    /// Apply RTL layout if current language is RTL
    func localizedLayout() -> some View {
        environment(\.layoutDirection,
                    LocalizationService.shared.currentLanguage.isRTL ? .rightToLeft : .leftT
    }
}

// MARK: - String Extension

extension String {
    /// Translate this key
    var localized: String {
        LocalizationService.shared.translate(self)
    }

    /// Translate with values
    func localized(with values: [String: Any]) -> String {
        LocalizationService.shared.translate(self, values: values)
    }
```

```
}
```

File: `ThinkTank/Views/Components/LocalizedText.swift`

```swift
// ============================================================================
// RADIANT v4.7.0 - LocalizedText SwiftUI Component
// ============================================================================

import SwiftUI

/// SwiftUI component for localized text
struct LocalizedText: View {
    @ObservedObject private var localization = LocalizationService.shared

    let key: String
    let values: [String: Any]

    init(_ key: String, values: [String: Any] = [:]) {
        self.key = key
        self.values = values
    }

    var body: some View {
        Text(localization.translate(key, values: values))
    }
}

/// Localized button with automatic translation
struct LocalizedButton: View {
    @ObservedObject private var localization = LocalizationService.shared

    let key: String
    let action: () -> Void

    init(_ key: String, action: @escaping () -> Void) {
        self.key = key
        self.action = action
    }

    var body: some View {
        Button(action: action) {
            Text(localization.translate(key))
        }
    }
}
```

```
/// Language selector view
struct LanguageSelector: View {
    @ObservedObject private var localization = LocalizationService.shared
    @State private var showingPicker = false

    var body: some View {
        Button {
            showingPicker = true
        } label: {
            HStack {
                Image(systemName: "globe")
                Text(localization.currentLanguage.displayName)
            }
        }
        .sheet(isPresented: $showingPicker) {
            NavigationStack {
                List(SupportedLanguage.allCases, id: \.self) { language in
                    Button {
                        Task {
                            await localization.setLanguage(language)
                        }
                        showingPicker = false
                    } label: {
                        HStack {
                            Text(language.displayName)
                            Spacer()
                            if language == localization.currentLanguage {
                                Image(systemName: "checkmark")
                                    .foregroundColor(.accentColor)
                            }
                        }
                    }
                    .foregroundColor(.primary)
                }
                .navigationTitle("ui.settings.language".localized)
                .toolbar {
                    ToolbarItem(placement: .cancellationAction) {
                        Button("ui.buttons.cancel".localized) {
                            showingPicker = false
                        }
                    }
                }
            }
        }
    }
}
```

```swift
// MARK: - Preview

#Preview {
    VStack(spacing: 20) {
        LocalizedText("ui.buttons.submit")
        LocalizedText("messages.welcome", values: ["name": "John"])
        LocalizedButton("ui.buttons.save") {
            print("Saved!")
        }
        LanguageSelector()
    }
    .padding()
}
```

41.9 ADMIN DASHBOARD - LOCALIZATION MANAGEMENT

File: `admin-dashboard/app/localization/page.tsx`

```tsx
// ============================================================================
// RADIANT v4.7.0 - Localization Management Dashboard
// ============================================================================

'use client';

import React, { useState, useEffect } from 'react';
import {
  Box,
  Typography,
  Tabs,
  Tab,
  Card,
  CardContent,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  Paper,
  Chip,
  IconButton,
  Button,
```

```
  TextField,
  Select,
  MenuItem,
  Dialog,
  DialogTitle,
  DialogContent,
  DialogActions,
  LinearProgress,
  Alert,
  Tooltip,
  Badge,
} from '@mui/material';
import {
  Edit,
  Check,
  Close,
  Refresh,
  Search,
  Language,
  Warning,
  CheckCircle,
  AutoAwesome,
} from '@mui/icons-material';
import { useTranslation } from '@/hooks/useTranslation';
import { api } from '@/lib/api';

interface Translation {
  id: string;
  registryId: string;
  languageCode: string;
  translatedText: string;
  status: 'pending' | 'ai_translated' | 'in_review' | 'approved' | 'rejected';
  aiModel?: string;
  aiConfidence?: number;
  reviewedBy?: string;
  reviewedAt?: string;
  key: string;
  defaultText: string;
  context?: string;
  category: string;
  languageName: string;
  nativeName: string;
}

interface TranslationCoverage {
  languageCode: string;
```

```
  languageName: string;
  totalStrings: number;
  translatedCount: number;
  approvedCount: number;
  aiTranslatedCount: number;
  pendingCount: number;
  coveragePercent: number;
}

export default function LocalizationPage() {
  const { t } = useTranslation();
  const [activeTab, setActiveTab] = useState(0);
  const [coverage, setCoverage] = useState<TranslationCoverage[]>([]);
  const [pendingReviews, setPendingReviews] = useState<Translation[]>([]);
  const [selectedLanguage, setSelectedLanguage] = useState<string>('all');
  const [searchQuery, setSearchQuery] = useState('');
  const [editDialog, setEditDialog] = useState<Translation | null>(null);
  const [editedText, setEditedText] = useState('');
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    loadData();
  }, [selectedLanguage]);

  async function loadData() {
    setLoading(true);
    try {
      const [coverageRes, pendingRes] = await Promise.all([
        api.get('/localization/coverage'),
        api.get('/localization/pending', {
          params: { language: selectedLanguage !== 'all' ? selectedLanguage : undefined }
        }),
      ]);
      setCoverage(coverageRes.data);
      setPendingReviews(pendingRes.data);
    } catch (error) {
      console.error('Failed to load localization data:', error);
    } finally {
      setLoading(false);
    }
  }

  async function handleApprove(translation: Translation) {
    try {
      await api.post(`/localization/translations/${translation.id}/approve`);
      loadData();
```

```
  } catch (error) {
    console.error('Failed to approve translation:', error);
  }
}

async function handleReject(translation: Translation, notes: string) {
  try {
    await api.post(`/localization/translations/${translation.id}/reject`, {
      reviewNotes: notes,
    });
    loadData();
  } catch (error) {
    console.error('Failed to reject translation:', error);
  }
}

async function handleSaveEdit() {
  if (!editDialog) return;
  try {
    await api.put(`/localization/translations/${editDialog.id}`, {
      translatedText: editedText,
      status: 'approved',
    });
    setEditDialog(null);
    loadData();
  } catch (error) {
    console.error('Failed to save translation:', error);
  }
}

async function handleBulkApprove() {
  const aiTranslated = pendingReviews.filter(t => t.status === 'ai_translated');
  if (aiTranslated.length === 0) return;

  if (!confirm(t('admin.localization.confirm_bulk_approve', { count: aiTranslated.length }
    return;
  }

  try {
    await api.post('/localization/bulk-approve', {
      translationIds: aiTranslated.map(t => t.id),
    });
    loadData();
  } catch (error) {
    console.error('Failed to bulk approve:', error);
  }
```

```
}

const getStatusChip = (status: string) => {
  const statusConfig = {
    pending: { color: 'default' as const, icon: <Warning fontSize="small" /> },
    ai_translated: { color: 'warning' as const, icon: <AutoAwesome fontSize="small" /> },
    in_review: { color: 'info' as const, icon: <Search fontSize="small" /> },
    approved: { color: 'success' as const, icon: <CheckCircle fontSize="small" /> },
    rejected: { color: 'error' as const, icon: <Close fontSize="small" /> },
  };
  const config = statusConfig[status as keyof typeof statusConfig] || statusConfig.pending
  return (
    <Chip
      size="small"
      label={t(`admin.localization.status.${status}`)}
      color={config.color}
      icon={config.icon}
    />
  );
};

const aiTranslatedCount = pendingReviews.filter(t => t.status === 'ai_translated').length

return (
  <Box sx={{ p: 3 }}>
    <Box sx={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center', mb
      <Typography variant="h4">
        <Language sx={{ mr: 1, verticalAlign: 'middle' }} />
        {t('admin.localization.title')}
      </Typography>
      <Button
        variant="outlined"
        startIcon={<Refresh />}
        onClick={loadData}
      >
        {t('admin.buttons.refresh')}
      </Button>
    </Box>

    {loading && <LinearProgress sx={{ mb: 2 }} />}

    <Tabs value={activeTab} onChange={(_, v) => setActiveTab(v)} sx={{ mb: 3 }}>
      <Tab label={t('admin.localization.tabs.coverage')} />
      <Tab
        label={
          <Badge badgeContent={aiTranslatedCount} color="warning">
```

```
        {t('admin.localization.tabs.review')}
      </Badge>
    }
  />
  <Tab label={t('admin.localization.tabs.registry')} />
</Tabs>

{/* Coverage Tab */}
{activeTab === 0 && (
  <TableContainer component={Paper}>
    <Table>
      <TableHead>
        <TableRow>
          <TableCell>{t('admin.localization.language')}</TableCell>
          <TableCell align="right">{t('admin.localization.total')}</TableCell>
          <TableCell align="right">{t('admin.localization.approved')}</TableCell>
          <TableCell align="right">{t('admin.localization.ai_translated')}</TableCell>
          <TableCell align="right">{t('admin.localization.pending')}</TableCell>
          <TableCell>{t('admin.localization.coverage')}</TableCell>
        </TableRow>
      </TableHead>
      <TableBody>
        {coverage.map((lang) => (
          <TableRow key={lang.languageCode}>
            <TableCell>
              <Box sx={{ display: 'flex', alignItems: 'center', gap: 1 }}>
                <Typography fontWeight="medium">{lang.languageName}</Typography>
                <Typography variant="caption" color="text.secondary">
                  ({lang.languageCode})
                </Typography>
              </Box>
            </TableCell>
            <TableCell align="right">{lang.totalStrings}</TableCell>
            <TableCell align="right">
              <Chip size="small" color="success" label={lang.approvedCount} />
            </TableCell>
            <TableCell align="right">
              {lang.aiTranslatedCount > 0 && (
                <Chip size="small" color="warning" label={lang.aiTranslatedCount} />
              )}
            </TableCell>
            <TableCell align="right">
              {lang.pendingCount > 0 && (
                <Chip size="small" color="default" label={lang.pendingCount} />
              )}
            </TableCell>
```

```
                <TableCell>
                  <Box sx={{ display: 'flex', alignItems: 'center', gap: 1 }}>
                    <LinearProgress
                      variant="determinate"
                      value={lang.coveragePercent}
                      sx={{ width: 100, height: 8, borderRadius: 4 }}
                      color={lang.coveragePercent === 100 ? 'success' : 'primary'}
                    />
                    <Typography variant="body2">
                      {lang.coveragePercent.toFixed(1)}%
                    </Typography>
                  </Box>
                </TableCell>
              </TableRow>
            ))}
          </TableBody>
        </Table>
      </TableContainer>
    )}

    {/* Review Tab */}
    {activeTab === 1 && (
      <Box>
        <Box sx={{ display: 'flex', gap: 2, mb: 2 }}>
          <Select
            size="small"
            value={selectedLanguage}
            onChange={(e) => setSelectedLanguage(e.target.value)}
            sx={{ minWidth: 200 }}
          >
            <MenuItem value="all">{t('admin.localization.all_languages')}</MenuItem>
            {coverage.map((lang) => (
              <MenuItem key={lang.languageCode} value={lang.languageCode}>
                {lang.languageName}
              </MenuItem>
            ))}
          </Select>
          <TextField
            size="small"
            placeholder={t('admin.localization.search_placeholder')}
            value={searchQuery}
            onChange={(e) => setSearchQuery(e.target.value)}
            InputProps={{ startAdornment: <Search sx={{ mr: 1, color: 'text.secondary' }}
          />
          {aiTranslatedCount > 0 && (
            <Button
```

```
      variant="contained"
      color="warning"
      startIcon={<CheckCircle />}
      onClick={handleBulkApprove}
    >
      {t('admin.localization.bulk_approve', { count: aiTranslatedCount })}
    </Button>
  )}
</Box>

{pendingReviews.length === 0 ? (
  <Alert severity="success">
    {t('admin.localization.no_pending_reviews')}
  </Alert>
) : (
  <TableContainer component={Paper}>
    <Table>
      <TableHead>
        <TableRow>
          <TableCell>{t('admin.localization.key')}</TableCell>
          <TableCell>{t('admin.localization.original')}</TableCell>
          <TableCell>{t('admin.localization.translation')}</TableCell>
          <TableCell>{t('admin.localization.language')}</TableCell>
          <TableCell>{t('admin.localization.status')}</TableCell>
          <TableCell align="right">{t('admin.localization.actions')}</TableCell>
        </TableRow>
      </TableHead>
      <TableBody>
        {pendingReviews
          .filter(t =>
            !searchQuery ||
            t.key.includes(searchQuery) ||
            t.defaultText.includes(searchQuery) ||
            t.translatedText.includes(searchQuery)
          )
          .map((translation) => (
            <TableRow key={translation.id}>
              <TableCell>
                <Typography variant="body2" fontFamily="monospace">
                  {translation.key}
                </Typography>
                <Typography variant="caption" color="text.secondary">
                  {translation.category}
                </Typography>
              </TableCell>
              <TableCell>
```

```
            <Typography variant="body2">
              {translation.defaultText}
            </Typography>
            {translation.context && (
              <Typography variant="caption" color="text.secondary">
                {translation.context}
              </Typography>
            )}
          </TableCell>
          <TableCell>
            <Typography
              variant="body2"
              dir={translation.languageCode === 'ar' ? 'rtl' : 'ltr'}
            >
              {translation.translatedText}
            </Typography>
            {translation.aiConfidence && (
              <Typography variant="caption" color="text.secondary">
                AI confidence: {(translation.aiConfidence * 100).toFixed(0)}%
              </Typography>
            )}
          </TableCell>
          <TableCell>
            <Chip
              size="small"
              label={translation.nativeName || translation.languageName}
            />
          </TableCell>
          <TableCell>
            {getStatusChip(translation.status)}
          </TableCell>
          <TableCell align="right">
            <Tooltip title={t('admin.buttons.edit')}>
              <IconButton
                size="small"
                onClick={() => {
                  setEditDialog(translation);
                  setEditedText(translation.translatedText);
                }}
              >
                <Edit fontSize="small" />
              </IconButton>
            </Tooltip>
            <Tooltip title={t('admin.buttons.approve')}>
              <IconButton
                size="small"
```

```
                        color="success"
                        onClick={() => handleApprove(translation)}
                      >
                        <Check fontSize="small" />
                      </IconButton>
                    </Tooltip>
                    <Tooltip title={t('admin.buttons.reject')}>
                      <IconButton
                        size="small"
                        color="error"
                        onClick={() => {
                          const notes = prompt(t('admin.localization.reject_reason'));
                          if (notes) handleReject(translation, notes);
                        }}
                      >
                        <Close fontSize="small" />
                      </IconButton>
                    </Tooltip>
                  </TableCell>
                </TableRow>
              ))}
            </TableBody>
          </Table>
        </TableContainer>
      )}
    </Box>
  )}

  {/* Edit Dialog */}
  <Dialog open={!!editDialog} onClose={() => setEditDialog(null)} maxWidth="md" fullWidt
    <DialogTitle>{t('admin.localization.edit_translation')}</DialogTitle>
    <DialogContent>
      {editDialog && (
        <Box sx={{ pt: 2 }}>
          <Typography variant="subtitle2" gutterBottom>
            {t('admin.localization.key')}: <code>{editDialog.key}</code>
          </Typography>
          <Typography variant="body2" color="text.secondary" gutterBottom>
            {t('admin.localization.original')}: {editDialog.defaultText}
          </Typography>
          {editDialog.context && (
            <Typography variant="body2" color="text.secondary" gutterBottom>
              {t('admin.localization.context')}: {editDialog.context}
            </Typography>
          )}
          <TextField
```

```
                fullWidth
                multiline
                rows={4}
                value={editedText}
                onChange={(e) => setEditedText(e.target.value)}
                label={t('admin.localization.translation')}
                sx={{ mt: 2 }}
                dir={editDialog.languageCode === 'ar' ? 'rtl' : 'ltr'}
              />
            </Box>
          )}
        </DialogContent>
        <DialogActions>
          <Button onClick={() => setEditDialog(null)}>
            {t('admin.buttons.cancel')}
          </Button>
          <Button onClick={handleSaveEdit} variant="contained" color="primary">
            {t('admin.buttons.save_and_approve')}
          </Button>
        </DialogActions>
      </Dialog>
    </Box>
  );
}
```

41.10 CDK INFRASTRUCTURE

File: `cdk/lib/localization-stack.ts`

```
// ============================================================================
// RADIANT v4.7.0 - Localization Infrastructure Stack
// ============================================================================

import * as cdk from 'aws-cdk-lib';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as sqs from 'aws-cdk-lib/aws-sqs';
import * as sns from 'aws-cdk-lib/aws-sns';
import * as events from 'aws-cdk-lib/aws-events';
import * as targets from 'aws-cdk-lib/aws-events-targets';
import * as iam from 'aws-cdk-lib/aws-iam';
import { SqsEventSource } from 'aws-cdk-lib/aws-lambda-event-sources';
import { Construct } from 'constructs';

interface LocalizationStackProps extends cdk.StackProps {
  environment: string;
```

```
  vpcId: string;
  dbSecurityGroupId: string;
  dbHost: string;
  dbName: string;
  dbSecretArn: string;
  adminNotificationTopicArn: string;
  adminUrl: string;
}

export class LocalizationStack extends cdk.Stack {
  public readonly translationQueue: sqs.Queue;
  public readonly translateLambda: lambda.Function;
  public readonly apiLambda: lambda.Function;

  constructor(scope: Construct, id: string, props: LocalizationStackProps) {
    super(scope, id, props);

    // Dead letter queue for failed translations
    const dlq = new sqs.Queue(this, 'TranslationDLQ', {
      queueName: `radiant-${props.environment}-translation-dlq`,
      retentionPeriod: cdk.Duration.days(14),
    });

    // Translation queue (FIFO for ordering by language)
    this.translationQueue = new sqs.Queue(this, 'TranslationQueue', {
      queueName: `radiant-${props.environment}-translation-queue.fifo`,
      fifo: true,
      contentBasedDeduplication: true,
      visibilityTimeout: cdk.Duration.minutes(5),
      deadLetterQueue: {
        queue: dlq,
        maxReceiveCount: 3,
      },
    });

    // Lambda execution role with Bedrock access
    const lambdaRole = new iam.Role(this, 'LocalizationLambdaRole', {
      assumedBy: new iam.ServicePrincipal('lambda.amazonaws.com'),
      managedPolicies: [
        iam.ManagedPolicy.fromAwsManagedPolicyName('service-role/AWSLambdaVPCAccessExecutio
      ],
    });

    // Bedrock permissions
    lambdaRole.addToPolicy(new iam.PolicyStatement({
      actions: [
```

```
      'bedrock:InvokeModel',
      'bedrock:InvokeModelWithResponseStream',
    ],
    resources: ['*'],
  }));

  // SNS permissions for notifications
  lambdaRole.addToPolicy(new iam.PolicyStatement({
    actions: ['sns:Publish'],
    resources: [props.adminNotificationTopicArn],
  }));

  // Secrets Manager for DB credentials
  lambdaRole.addToPolicy(new iam.PolicyStatement({
    actions: ['secretsmanager:GetSecretValue'],
    resources: [props.dbSecretArn],
  }));

  // Common Lambda environment - use DATABASE_URL for consistency
  const lambdaEnvironment = {
    NODE_ENV: props.environment,
    DATABASE_URL: props.databaseUrl,  // Connection string format
    ADMIN_NOTIFICATION_TOPIC_ARN: props.adminNotificationTopicArn,
    ADMIN_URL: props.adminUrl,
    TRANSLATION_QUEUE_URL: this.translationQueue.queueUrl,
  };

  // Translation Lambda (processes queue)
  this.translateLambda = new lambda.Function(this, 'TranslateLambda', {
    functionName: `radiant-${props.environment}-localization-translate`,
    runtime: lambda.Runtime.NODEJS_20_X,
    handler: 'translate.handler',
    code: lambda.Code.fromAsset('lambda/localization'),
    timeout: cdk.Duration.minutes(2),
    memorySize: 512,
    role: lambdaRole,
    environment: lambdaEnvironment,
  });

  // Add SQS trigger
  this.translateLambda.addEventSource(new SqsEventSource(this.translationQueue, {
    batchSize: 1,
    maxConcurrency: 10,
  }));

  // Queue processor Lambda (scheduled)
```

```
    const queueProcessorLambda = new lambda.Function(this, 'QueueProcessorLambda', {
      functionName: `radiant-${props.environment}-localization-queue-processor`,
      runtime: lambda.Runtime.NODEJS_20_X,
      handler: 'process-queue.handler',
      code: lambda.Code.fromAsset('lambda/localization'),
      timeout: cdk.Duration.minutes(1),
      memorySize: 256,
      role: lambdaRole,
      environment: lambdaEnvironment,
    });

    // Grant queue send permissions
    this.translationQueue.grantSendMessages(queueProcessorLambda);

    // Schedule queue processor every 5 minutes
    new events.Rule(this, 'QueueProcessorSchedule', {
      ruleName: `radiant-${props.environment}-translation-queue-processor`,
      schedule: events.Schedule.rate(cdk.Duration.minutes(5)),
      targets: [new targets.LambdaFunction(queueProcessorLambda)],
    });

    // API Lambda
    this.apiLambda = new lambda.Function(this, 'LocalizationApiLambda', {
      functionName: `radiant-${props.environment}-localization-api`,
      runtime: lambda.Runtime.NODEJS_20_X,
      handler: 'api.handler',
      code: lambda.Code.fromAsset('lambda/localization'),
      timeout: cdk.Duration.seconds(30),
      memorySize: 512,
      role: lambdaRole,
      environment: lambdaEnvironment,
    });

    // Outputs
    new cdk.CfnOutput(this, 'TranslationQueueUrl', {
      value: this.translationQueue.queueUrl,
      exportName: `radiant-${props.environment}-translation-queue-url`,
    });

    new cdk.CfnOutput(this, 'LocalizationApiArn', {
      value: this.apiLambda.functionArn,
      exportName: `radiant-${props.environment}-localization-api-arn`,
    });
  }
}
```

41.11 INITIAL TRANSLATION SEED DATA

Migration: 041b_seed_localization.sql

```
-- ============================================================================
-- RADIANT v4.7.0 - Seed Initial Translations
-- Migration: 041b_seed_localization.sql
-- ============================================================================

-- UI Buttons
INSERT INTO localization_registry (key, default_text, category, context, source_app) VALUES
('ui.buttons.submit', 'Submit', 'ui.buttons', 'Primary form submission button', 'shared'),
('ui.buttons.cancel', 'Cancel', 'ui.buttons', 'Cancel/close action', 'shared'),
('ui.buttons.save', 'Save', 'ui.buttons', 'Save changes button', 'shared'),
('ui.buttons.delete', 'Delete', 'ui.buttons', 'Delete/remove action', 'shared'),
('ui.buttons.edit', 'Edit', 'ui.buttons', 'Edit/modify action', 'shared'),
('ui.buttons.close', 'Close', 'ui.buttons', 'Close dialog/modal', 'shared'),
('ui.buttons.confirm', 'Confirm', 'ui.buttons', 'Confirmation action', 'shared'),
('ui.buttons.back', 'Back', 'ui.buttons', 'Navigate back', 'shared'),
('ui.buttons.next', 'Next', 'ui.buttons', 'Navigate forward/next step', 'shared'),
('ui.buttons.refresh', 'Refresh', 'ui.buttons', 'Reload/refresh data', 'shared'),
('ui.buttons.search', 'Search', 'ui.buttons', 'Search action', 'shared'),
('ui.buttons.send', 'Send', 'ui.buttons', 'Send message/request', 'shared'),
('ui.buttons.copy', 'Copy', 'ui.buttons', 'Copy to clipboard', 'shared'),
('ui.buttons.download', 'Download', 'ui.buttons', 'Download file', 'shared'),
('ui.buttons.upload', 'Upload', 'ui.buttons', 'Upload file', 'shared'),
('ui.buttons.retry', 'Retry', 'ui.buttons', 'Retry failed action', 'shared'),
('ui.buttons.approve', 'Approve', 'ui.buttons', 'Approve action (admin)', 'admin'),
('ui.buttons.reject', 'Reject', 'ui.buttons', 'Reject action (admin)', 'admin'),
('ui.buttons.save_and_approve', 'Save & Approve', 'ui.buttons', 'Save and approve translatio

-- Messages
INSERT INTO localization_registry (key, default_text, category, context, source_app, placeho
('messages.welcome', 'Welcome, {name}!', 'messages.success', 'Welcome message after login',
('messages.saved', 'Changes saved successfully', 'messages.success', 'After successful save'
('messages.deleted', 'Item deleted successfully', 'messages.success', 'After successful dele
('messages.copied', 'Copied to clipboard', 'messages.success', 'After copy action', 'shared'
('messages.loading', 'Loading...', 'messages.loading', 'Generic loading state', 'shared', '
('messages.processing', 'Processing...', 'messages.loading', 'Processing action', 'shared',
('messages.no_results', 'No results found', 'messages.info', 'Empty search results', 'shared
('messages.confirm_delete', 'Are you sure you want to delete this item?', 'messages.warning'

-- Errors
INSERT INTO localization_registry (key, default_text, category, context, source_app, placeho
```

```
('errors.generic', 'An error occurred. Please try again.', 'errors.system', 'Generic error f
('errors.network', 'Network error. Please check your connection.', 'errors.network', 'Networ
('errors.unauthorized', 'You are not authorized to perform this action.', 'errors.auth', 'Pe
('errors.session_expired', 'Your session has expired. Please log in again.', 'errors.auth',
('errors.not_found', 'The requested item was not found.', 'errors.api', '404 error', 'shared
('errors.validation', 'Please check your input and try again.', 'errors.validation', 'Form v
('errors.required_field', 'This field is required.', 'errors.validation', 'Required field va
('errors.invalid_email', 'Please enter a valid email address.', 'errors.validation', 'Email
('errors.rate_limit', 'Too many requests. Please wait a moment.', 'errors.api', 'Rate limiti

-- Think Tank specific
INSERT INTO localization_registry (key, default_text, category, context, source_app, placeho
('thinktank.chat.placeholder', 'Ask me anything...', 'features.thinktank', 'Chat input place
('thinktank.chat.thinking', 'Thinking...', 'features.thinktank', 'AI is processing', 'thinkt
('thinktank.chat.error', 'Sorry, I encountered an error. Please try again.', 'features.think
('thinktank.chat.regenerate', 'Regenerate response', 'features.thinktank', 'Button to regene
('thinktank.chat.stop', 'Stop generating', 'features.thinktank', 'Button to stop AI generati
('thinktank.chat.new', 'New conversation', 'features.thinktank', 'Start new chat', 'thinktan
('thinktank.models.select', 'Select model', 'features.thinktank', 'Model selector label', 't
('thinktank.models.auto', 'Auto (recommended)', 'features.thinktank', 'Automatic model selec

-- Billing & Subscription Tiers (v4.13.0+)
INSERT INTO localization_registry (key, default_text, category, context, source_app) VALUES
('billing.tiers.free.name', 'Free Trial', 'billing.tiers', 'Free tier display name', 'shared
('billing.tiers.free.description', 'Try RADIANT with limited features', 'billing.tiers', 'Fr
('billing.tiers.individual.name', 'Individual', 'billing.tiers', 'Individual tier display na
('billing.tiers.individual.description', 'Full-featured personal plan', 'billing.tiers', 'In
('billing.tiers.family.name', 'Family', 'billing.tiers', 'Family tier display name', 'shared
('billing.tiers.family.description', 'Share AI across your household', 'billing.tiers', 'Fam
('billing.tiers.team.name', 'Team', 'billing.tiers', 'Team tier display name', 'shared'),
('billing.tiers.team.description', 'Collaborate with your small team', 'billing.tiers', 'Tea
('billing.tiers.team.badge', 'Most Popular', 'billing.badges', 'Team tier badge text', 'shar
('billing.tiers.business.name', 'Business', 'billing.tiers', 'Business tier display name',
('billing.tiers.business.description', 'Enterprise features for growing companies', 'billing
('billing.tiers.enterprise.name', 'Enterprise', 'billing.tiers', 'Enterprise tier display na
('billing.tiers.enterprise.description', 'Full-scale enterprise deployment', 'billing.tiers'
('billing.tiers.enterprise_plus.name', 'Enterprise Plus', 'billing.tiers', 'Enterprise Plus
('billing.tiers.enterprise_plus.description', 'Maximum security and compliance', 'billing.ti
('billing.tiers.enterprise_plus.badge', 'Full Compliance Included', 'billing.badges', 'Enter
('billing.credits.title', 'Credits', 'billing.credits', 'Credits section title', 'shared'),
('billing.credits.balance', 'Credit Balance', 'billing.credits', 'Current credit balance lab
('billing.credits.purchase', 'Purchase Credits', 'billing.credits', 'Buy credits button', 's
('billing.credits.low_balance', 'Low credit balance', 'billing.credits', 'Low balance warnin

-- Admin specific
INSERT INTO localization_registry (key, default_text, category, context, source_app) VALUES
```

```
('admin.nav.dashboard', 'Dashboard', 'features.admin', 'Navigation item', 'admin'),
('admin.nav.users', 'Users', 'features.admin', 'Navigation item', 'admin'),
('admin.nav.tenants', 'Tenants', 'features.admin', 'Navigation item', 'admin'),
('admin.nav.models', 'AI Models', 'features.admin', 'Navigation item', 'admin'),
('admin.nav.billing', 'Billing', 'features.admin', 'Navigation item', 'admin'),
('admin.nav.localization', 'Localization', 'features.admin', 'Navigation item', 'admin'),
('admin.nav.settings', 'Settings', 'features.admin', 'Navigation item', 'admin');

-- Localization admin specific
INSERT INTO localization_registry (key, default_text, category, context, source_app, placeho
('admin.localization.title', 'Localization Management', 'features.admin', 'Page title', 'adm
('admin.localization.tabs.coverage', 'Coverage', 'features.admin', 'Tab label', 'admin', '[
('admin.localization.tabs.review', 'Review Queue', 'features.admin', 'Tab label', 'admin',
('admin.localization.tabs.registry', 'String Registry', 'features.admin', 'Tab label', 'admi
('admin.localization.language', 'Language', 'features.admin', 'Table header', 'admin', '[]'
('admin.localization.total', 'Total', 'features.admin', 'Table header', 'admin', '[]'),
('admin.localization.approved', 'Approved', 'features.admin', 'Table header', 'admin', '[]'
('admin.localization.ai_translated', 'AI Translated', 'features.admin', 'Table header', 'adm
('admin.localization.pending', 'Pending', 'features.admin', 'Table header', 'admin', '[]'),
('admin.localization.coverage', 'Coverage', 'features.admin', 'Table header', 'admin', '[]'
('admin.localization.key', 'Key', 'features.admin', 'Table header', 'admin', '[]'),
('admin.localization.original', 'Original (English)', 'features.admin', 'Table header', 'adm
('admin.localization.translation', 'Translation', 'features.admin', 'Table header/label', 'a
('admin.localization.status', 'Status', 'features.admin', 'Table header', 'admin', '[]'),
('admin.localization.actions', 'Actions', 'features.admin', 'Table header', 'admin', '[]'),
('admin.localization.context', 'Context', 'features.admin', 'Context for translators', 'admi
('admin.localization.all_languages', 'All Languages', 'features.admin', 'Filter option', 'ad
('admin.localization.search_placeholder', 'Search keys or text...', 'features.admin', 'Searc
('admin.localization.no_pending_reviews', 'No translations pending review. Great work!', 'fe
('admin.localization.edit_translation', 'Edit Translation', 'features.admin', 'Dialog title'
('admin.localization.reject_reason', 'Please provide a reason for rejection:', 'features.adm
('admin.localization.bulk_approve', 'Approve All AI Translations ({count})', 'features.admin
('admin.localization.confirm_bulk_approve', 'Approve {count} AI-translated strings?', 'featu
('admin.localization.status.pending', 'Pending', 'features.admin', 'Status label', 'admin',
('admin.localization.status.ai_translated', 'AI Translated', 'features.admin', 'Status label
('admin.localization.status.in_review', 'In Review', 'features.admin', 'Status label', 'admi
('admin.localization.status.approved', 'Approved', 'features.admin', 'Status label', 'admin'
('admin.localization.status.rejected', 'Rejected', 'features.admin', 'Status label', 'admin'

-- Settings
INSERT INTO localization_registry (key, default_text, category, context, source_app) VALUES
('ui.settings.language', 'Language', 'features.settings', 'Language setting label', 'shared'
('ui.settings.theme', 'Theme', 'features.settings', 'Theme setting label', 'shared'),
('ui.settings.notifications', 'Notifications', 'features.settings', 'Notifications setting l
('ui.settings.profile', 'Profile', 'features.settings', 'Profile section label', 'shared'),
('ui.settings.security', 'Security', 'features.settings', 'Security section label', 'shared'
```

```
('ui.settings.account', 'Account', 'features.settings', 'Account section label', 'shared');

-- Note: English translations are auto-created by the trigger from default_text
-- Other languages will be AI-translated automatically via the queue
```

SECTION-42-DYNAMIC-CONFIGURATION

SECTION 42: DYNAMIC CONFIGURATION MANAGEMENT SYSTEM (v4.8.0)

> **CRITICAL: This section eliminates ALL hardcoded runtime parameters. Every configurable value is now stored in the database and editable by admins.**

42.1 ARCHITECTURE OVERVIEW

The Problem

Before v4.8.0, RADIANT had numerous hardcoded parameters scattered throughout the codebase:

```
BEFORE (v4.7.x and earlier):

 Hardcoded Parameters Everywhere


  TypeScript Constants:
  const MAX_TOKENS = 128000;  ← Hardcoded!
  const DEFAULT_MARGIN = 0.40;  ← Hardcoded!
  const RETRY_ATTEMPTS = 3;  ← Hardcoded!

  Lambda Environment:
  timeout: Duration.seconds(30)  ← Hardcoded!
  memorySize: 512  ← Hardcoded!

  Rate Limits:
```

```
  maxConcurrent: 10  ← Hardcoded!
  maxPerMinute: 100  ← Hardcoded!

  Problems:
     Cannot adjust without code deployment
     No per-tenant customization
     No audit trail of changes
     Incident response requires developer
```

The Solution

v4.8.0 implements **complete database-driven configuration**:

```
AFTER (v4.8.0):

 Centralized Configuration Registry



    system_configuration (Global Defaults)

     key: "pricing.external_provider_margin"
     value: 0.40
     category: "pricing"
     type: "decimal"
     min: 0.00, max: 1.00




    tenant_configuration_overrides
       tenant_abc: 0.35 (negotiated discount)
       tenant_xyz: 0.50 (premium support)
       (others use global default 0.40)


  Usage: getConfig('pricing.external_provider_margin', tenantId)
  Returns: 0.35 for tenant_abc, 0.40 for others
```

Configuration Categories (12)

Category	Description	Example Parameters
`rate_limits`	API and service rate limits	requests_per_minute, concurrent_connections
`timeouts`	Request and processing timeouts	lambda_timeout, request_timeout, session_idle
`pricing`	Margins, markups, discounts	external_margin, self_hosted_margin, minimum_charge
`tokens`	Token and context limits	max_tokens_per_request, max_context_window
`retry`	Retry and backoff configuration	max_attempts, initial_delay, backoff_multiplier
`cache`	Cache TTL and invalidation	translation_bundle_ttl, model_list_ttl
`thresholds`	Health, confidence, quality thresholds	health_check_threshold, confidence_minimum
`discounts`	Volume discount tiers	tier_1_threshold, tier_1_discount
`session`	Auth and session settings	token_expiry, refresh_window, max_sessions
`translation`	i18n system settings	concurrent_limit, per_minute_limit
`workflow`	Workflow proposal thresholds	min_occurrences, min_unique_users
`notifications`	Alert and notification settings	alert_thresholds, channels, cooldown

---

42.2 DATABASE SCHEMA

Migration: 042_configuration_management.sql

```sql
-- ============================================================================
-- RADIANT v4.8.0 - Dynamic Configuration Management System
-- Migration: 042_configuration_management.sql
-- ============================================================================

-- ============================================================================
-- 42.2.1 Configuration Categories
-- ============================================================================

CREATE TABLE configuration_categories (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
```

```
    description TEXT,
    display_order INTEGER DEFAULT 100,
    icon VARCHAR(50),  -- Material UI icon name
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

INSERT INTO configuration_categories (id, name, description, display_order, icon) VALUES
('rate_limits', 'Rate Limits', 'API and service rate limiting configuration', 1, 'Speed'),
('timeouts', 'Timeouts', 'Request and processing timeout settings', 2, 'Timer'),
('pricing', 'Pricing', 'Margins, markups, and discount configuration', 3, 'AttachMoney'),
('tokens', 'Token Limits', 'Token and context window limits', 4, 'Token'),
('retry', 'Retry Configuration', 'Retry attempts and backoff settings', 5, 'Refresh'),
('cache', 'Cache Settings', 'Cache TTL and invalidation rules', 6, 'Cached'),
('thresholds', 'Thresholds', 'Health, confidence, and quality thresholds', 7, 'TrendingUp'),
('discounts', 'Volume Discounts', 'Volume-based discount tier configuration', 8, 'Discount'),
('session', 'Session & Auth', 'Authentication and session settings', 9, 'Lock'),
('translation', 'Translation System', 'i18n and translation service settings', 10, 'Translat
('workflow', 'Workflow Proposals', 'Dynamic workflow proposal thresholds', 11, 'AccountTree'
('notifications', 'Notifications', 'Alert thresholds and notification channels', 12, 'Notifi

-- ============================================================================
-- 42.2.2 Configuration Value Types
-- ============================================================================

CREATE TYPE config_value_type AS ENUM (
    'string',
    'integer',
    'decimal',
    'boolean',
    'json',
    'duration',     -- Stored as seconds, displayed as human-readable
    'percentage',   -- Stored as decimal (0.40 = 40%)
    'enum'
);

-- ============================================================================
-- 42.2.3 System Configuration (Global Defaults)
-- ============================================================================

CREATE TABLE system_configuration (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Identification
    key VARCHAR(100) NOT NULL UNIQUE,
    category_id VARCHAR(50) NOT NULL REFERENCES configuration_categories(id),
```

```
    -- Value storage
    value_type config_value_type NOT NULL,
    value_string TEXT,           -- For string, enum types
    value_integer BIGINT,        -- For integer, duration types
    value_decimal DECIMAL(20,6), -- For decimal, percentage types
    value_boolean BOOLEAN,       -- For boolean type
    value_json JSONB,            -- For json type

    -- Metadata
    display_name VARCHAR(200) NOT NULL,
    description TEXT,
    unit VARCHAR(50),            -- e.g., 'seconds', 'requests', '%', 'tokens'

    -- Validation constraints
    min_value DECIMAL(20,6),
    max_value DECIMAL(20,6),
    enum_values TEXT[],          -- For enum type
    regex_pattern TEXT,          -- For string validation

    -- Environment scoping
    environment VARCHAR(20) DEFAULT 'all',  -- 'all', 'dev', 'staging', 'prod'

    -- Feature flags
    is_sensitive BOOLEAN DEFAULT FALSE,     -- Mask value in UI
    requires_restart BOOLEAN DEFAULT FALSE, -- Warn admin
    is_deprecated BOOLEAN DEFAULT FALSE,
    deprecated_replacement_key VARCHAR(100),

    -- Audit
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_by UUID REFERENCES administrators(id),

    -- Constraints
    CONSTRAINT valid_value CHECK (
        (value_type = 'string' AND value_string IS NOT NULL) OR
        (value_type = 'integer' AND value_integer IS NOT NULL) OR
        (value_type = 'decimal' AND value_decimal IS NOT NULL) OR
        (value_type = 'boolean' AND value_boolean IS NOT NULL) OR
        (value_type = 'json' AND value_json IS NOT NULL) OR
        (value_type = 'duration' AND value_integer IS NOT NULL) OR
        (value_type = 'percentage' AND value_decimal IS NOT NULL) OR
        (value_type = 'enum' AND value_string IS NOT NULL)
    )
);
```

```sql
CREATE INDEX idx_system_config_category ON system_configuration(category_id);
CREATE INDEX idx_system_config_key ON system_configuration(key);
CREATE INDEX idx_system_config_env ON system_configuration(environment);

-- ============================================================================
-- 42.2.4 Tenant Configuration Overrides
-- ============================================================================

CREATE TABLE tenant_configuration_overrides (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    config_id UUID NOT NULL REFERENCES system_configuration(id) ON DELETE CASCADE,

    -- Override value (same structure as system_configuration)
    value_string TEXT,
    value_integer BIGINT,
    value_decimal DECIMAL(20,6),
    value_boolean BOOLEAN,
    value_json JSONB,

    -- Validity period (optional)
    valid_from TIMESTAMPTZ DEFAULT NOW(),
    valid_until TIMESTAMPTZ,  -- NULL = forever

    -- Audit
    reason TEXT,              -- Why this override exists
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    created_by UUID REFERENCES administrators(id),
    updated_by UUID REFERENCES administrators(id),

    UNIQUE(tenant_id, config_id)
);

CREATE INDEX idx_tenant_config_tenant ON tenant_configuration_overrides(tenant_id);
CREATE INDEX idx_tenant_config_config ON tenant_configuration_overrides(config_id);
CREATE INDEX idx_tenant_config_validity ON tenant_configuration_overrides(valid_from, valid_

-- RLS
ALTER TABLE tenant_configuration_overrides ENABLE ROW LEVEL SECURITY;

CREATE POLICY tenant_config_isolation ON tenant_configuration_overrides
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

-- ============================================================================
-- 42.2.5 Configuration Audit Log
```

```sql
-- ============================================================================

CREATE TABLE configuration_audit_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- What changed
    config_id UUID REFERENCES system_configuration(id) ON DELETE SET NULL,
    tenant_override_id UUID REFERENCES tenant_configuration_overrides(id) ON DELETE SET NULL
    config_key VARCHAR(100) NOT NULL,
    tenant_id UUID,  -- NULL for global changes

    -- Change details
    action VARCHAR(50) NOT NULL,  -- 'created', 'updated', 'deleted', 'override_created', '
    old_value JSONB,
    new_value JSONB,

    -- Who made the change
    changed_by UUID REFERENCES administrators(id),
    changed_by_email VARCHAR(255),

    -- Context
    reason TEXT,
    ip_address INET,
    user_agent TEXT,

    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_config_audit_config ON configuration_audit_log(config_id);
CREATE INDEX idx_config_audit_tenant ON configuration_audit_log(tenant_id);
CREATE INDEX idx_config_audit_created ON configuration_audit_log(created_at DESC);
CREATE INDEX idx_config_audit_key ON configuration_audit_log(config_key);

-- ============================================================================
-- 42.2.6 Configuration Cache Invalidation
-- ============================================================================

CREATE TABLE configuration_cache_invalidation (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    config_key VARCHAR(100) NOT NULL,
    tenant_id UUID,  -- NULL = global invalidation
    invalidated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    processed_at TIMESTAMPTZ,
    processed_by VARCHAR(100)  -- Lambda function name that processed it
);
```

```
CREATE INDEX idx_config_cache_pending ON configuration_cache_invalidation(invalidated_at)
    WHERE processed_at IS NULL;

-- ============================================================================
-- 42.2.7 Helper Functions
-- ============================================================================

-- Get configuration value with tenant override support
CREATE OR REPLACE FUNCTION get_config(
    p_key VARCHAR(100),
    p_tenant_id UUID DEFAULT NULL,
    p_environment VARCHAR(20) DEFAULT 'prod'
) RETURNS JSONB AS $$
DECLARE
    v_config system_configuration%ROWTYPE;
    v_override tenant_configuration_overrides%ROWTYPE;
    v_result JSONB;
BEGIN
    -- Get base configuration
    SELECT * INTO v_config
    FROM system_configuration
    WHERE key = p_key
      AND (environment = 'all' OR environment = p_environment);

    IF v_config IS NULL THEN
        RETURN NULL;
    END IF;

    -- Build base result
    v_result = jsonb_build_object(
        'key', v_config.key,
        'type', v_config.value_type,
        'value', CASE v_config.value_type
            WHEN 'string' THEN to_jsonb(v_config.value_string)
            WHEN 'integer' THEN to_jsonb(v_config.value_integer)
            WHEN 'decimal' THEN to_jsonb(v_config.value_decimal)
            WHEN 'boolean' THEN to_jsonb(v_config.value_boolean)
            WHEN 'json' THEN v_config.value_json
            WHEN 'duration' THEN to_jsonb(v_config.value_integer)
            WHEN 'percentage' THEN to_jsonb(v_config.value_decimal)
            WHEN 'enum' THEN to_jsonb(v_config.value_string)
        END,
        'is_override', false
    );
```

```
    -- Check for tenant override if tenant_id provided
    IF p_tenant_id IS NOT NULL THEN
        SELECT * INTO v_override
        FROM tenant_configuration_overrides
        WHERE config_id = v_config.id
          AND tenant_id = p_tenant_id
          AND valid_from <= NOW()
          AND (valid_until IS NULL OR valid_until > NOW());

        IF v_override IS NOT NULL THEN
            v_result = jsonb_set(v_result, '{value}',
                CASE v_config.value_type
                    WHEN 'string' THEN to_jsonb(v_override.value_string)
                    WHEN 'integer' THEN to_jsonb(v_override.value_integer)
                    WHEN 'decimal' THEN to_jsonb(v_override.value_decimal)
                    WHEN 'boolean' THEN to_jsonb(v_override.value_boolean)
                    WHEN 'json' THEN v_override.value_json
                    WHEN 'duration' THEN to_jsonb(v_override.value_integer)
                    WHEN 'percentage' THEN to_jsonb(v_override.value_decimal)
                    WHEN 'enum' THEN to_jsonb(v_override.value_string)
                END
            );
            v_result = jsonb_set(v_result, '{is_override}', 'true'::jsonb);
        END IF;
    END IF;

    RETURN v_result;
END;
$$ LANGUAGE plpgsql STABLE;

-- Get all configurations for a category
CREATE OR REPLACE FUNCTION get_configs_by_category(
    p_category VARCHAR(50),
    p_tenant_id UUID DEFAULT NULL,
    p_environment VARCHAR(20) DEFAULT 'prod'
) RETURNS TABLE (
    key VARCHAR(100),
    value JSONB,
    display_name VARCHAR(200),
    description TEXT,
    value_type config_value_type,
    unit VARCHAR(50),
    is_override BOOLEAN
) AS $$
BEGIN
    RETURN QUERY
```

```
    SELECT
        sc.key,
        (get_config(sc.key, p_tenant_id, p_environment))->>'value',
        sc.display_name,
        sc.description,
        sc.value_type,
        sc.unit,
        ((get_config(sc.key, p_tenant_id, p_environment))->>'is_override')::BOOLEAN
    FROM system_configuration sc
    WHERE sc.category_id = p_category
      AND (sc.environment = 'all' OR sc.environment = p_environment)
      AND sc.is_deprecated = false
    ORDER BY sc.key;
END;
$$ LANGUAGE plpgsql STABLE;

-- Trigger to log configuration changes
CREATE OR REPLACE FUNCTION log_config_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO configuration_audit_log (
            config_id, config_key, action, new_value, changed_by
        ) VALUES (
            NEW.id, NEW.key, 'created',
            jsonb_build_object('value', COALESCE(
                to_jsonb(NEW.value_string),
                to_jsonb(NEW.value_integer),
                to_jsonb(NEW.value_decimal),
                to_jsonb(NEW.value_boolean),
                NEW.value_json
            )),
            NEW.updated_by
        );
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO configuration_audit_log (
            config_id, config_key, action, old_value, new_value, changed_by
        ) VALUES (
            NEW.id, NEW.key, 'updated',
            jsonb_build_object('value', COALESCE(
                to_jsonb(OLD.value_string),
                to_jsonb(OLD.value_integer),
                to_jsonb(OLD.value_decimal),
                to_jsonb(OLD.value_boolean),
                OLD.value_json
            )),
```

```
            jsonb_build_object('value', COALESCE(
                to_jsonb(NEW.value_string),
                to_jsonb(NEW.value_integer),
                to_jsonb(NEW.value_decimal),
                to_jsonb(NEW.value_boolean),
                NEW.value_json
            )),
            NEW.updated_by
        );

        -- Queue cache invalidation
        INSERT INTO configuration_cache_invalidation (config_key)
        VALUES (NEW.key);
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_config_changes
    AFTER INSERT OR UPDATE ON system_configuration
    FOR EACH ROW
    EXECUTE FUNCTION log_config_changes();

-- Trigger for tenant override changes
CREATE OR REPLACE FUNCTION log_tenant_override_changes()
RETURNS TRIGGER AS $$
DECLARE
    v_config_key VARCHAR(100);
BEGIN
    SELECT key INTO v_config_key FROM system_configuration WHERE id = COALESCE(NEW.config_id

    IF TG_OP = 'INSERT' THEN
        INSERT INTO configuration_audit_log (
            tenant_override_id, config_key, tenant_id, action, new_value, changed_by, reason
        ) VALUES (
            NEW.id, v_config_key, NEW.tenant_id, 'override_created',
            jsonb_build_object('value', COALESCE(
                to_jsonb(NEW.value_string),
                to_jsonb(NEW.value_integer),
                to_jsonb(NEW.value_decimal),
                to_jsonb(NEW.value_boolean),
                NEW.value_json
            )),
            NEW.created_by, NEW.reason
        );
```

```
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO configuration_audit_log (
            tenant_override_id, config_key, tenant_id, action, old_value, new_value, changed
        ) VALUES (
            NEW.id, v_config_key, NEW.tenant_id, 'override_updated',
            jsonb_build_object('value', COALESCE(
                to_jsonb(OLD.value_string),
                to_jsonb(OLD.value_integer),
                to_jsonb(OLD.value_decimal),
                to_jsonb(OLD.value_boolean),
                OLD.value_json
            )),
            jsonb_build_object('value', COALESCE(
                to_jsonb(NEW.value_string),
                to_jsonb(NEW.value_integer),
                to_jsonb(NEW.value_decimal),
                to_jsonb(NEW.value_boolean),
                NEW.value_json
            )),
            NEW.updated_by, NEW.reason
        );
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO configuration_audit_log (
            config_key, tenant_id, action, old_value, changed_by
        ) VALUES (
            v_config_key, OLD.tenant_id, 'override_deleted',
            jsonb_build_object('value', COALESCE(
                to_jsonb(OLD.value_string),
                to_jsonb(OLD.value_integer),
                to_jsonb(OLD.value_decimal),
                to_jsonb(OLD.value_boolean),
                OLD.value_json
            )),
            current_setting('app.current_admin_id', true)::UUID
        );
    END IF;

    -- Queue cache invalidation
    INSERT INTO configuration_cache_invalidation (config_key, tenant_id)
    VALUES (v_config_key, COALESCE(NEW.tenant_id, OLD.tenant_id));

    RETURN COALESCE(NEW, OLD);
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_tenant_override_changes
```

```
AFTER INSERT OR UPDATE OR DELETE ON tenant_configuration_overrides
FOR EACH ROW
EXECUTE FUNCTION log_tenant_override_changes();
```

42.3 SEED CONFIGURATION DATA

Migration: 042b_seed_configuration.sql

```
-- ============================================================================
-- RADIANT v4.8.0 - Seed Initial Configuration Values
-- Migration: 042b_seed_configuration.sql
-- ============================================================================

-- ============================================================================
-- Rate Limits
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('rate_limits.api.requests_per_minute', 'rate_limits', 'integer', 1000, 'API Requests per Mi
('rate_limits.api.requests_per_hour', 'rate_limits', 'integer', 50000, 'API Requests per Hou
('rate_limits.api.concurrent_connections', 'rate_limits', 'integer', 100, 'Concurrent Connec
('rate_limits.chat.messages_per_minute', 'rate_limits', 'integer', 60, 'Chat Messages per Mi
('rate_limits.chat.concurrent_sessions', 'rate_limits', 'integer', 5, 'Concurrent Chat Sessi
('rate_limits.file_upload.max_size_mb', 'rate_limits', 'integer', 100, 'Max File Upload Size
('rate_limits.file_upload.per_hour', 'rate_limits', 'integer', 50, 'File Uploads per Hour',

-- ============================================================================
-- Timeouts
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('timeouts.lambda.default', 'timeouts', 'duration', 30, 'Default Lambda Timeout', 'Default t
('timeouts.lambda.chat', 'timeouts', 'duration', 120, 'Chat Lambda Timeout', 'Timeout for ch
('timeouts.lambda.orchestration', 'timeouts', 'duration', 300, 'Orchestration Lambda Timeout
('timeouts.lambda.batch', 'timeouts', 'duration', 900, 'Batch Processing Timeout', 'Timeout
('timeouts.request.api_gateway', 'timeouts', 'duration', 29, 'API Gateway Timeout', 'API Gat
('timeouts.request.external_provider', 'timeouts', 'duration', 60, 'External Provider Timeou
('timeouts.session.idle', 'timeouts', 'duration', 1800, 'Session Idle Timeout', 'Time before
('timeouts.session.absolute', 'timeouts', 'duration', 86400, 'Absolute Session Timeout', 'Ma

-- ============================================================================
-- Pricing
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name,
```

```
('pricing.external_provider_margin', 'pricing', 'percentage', 0.40, 'External Provider Margi
('pricing.self_hosted_margin', 'pricing', 'percentage', 0.75, 'Self-Hosted Margin', 'Default
('pricing.minimum_charge', 'pricing', 'decimal', 0.01, 'Minimum Charge', 'Minimum charge per
('pricing.tax_rate_default', 'pricing', 'percentage', 0.00, 'Default Tax Rate', 'Default tax

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('pricing.invoice.due_days', 'pricing', 'integer', 30, 'Invoice Due Days', 'Days until invoi
('pricing.invoice.reminder_days', 'pricing', 'integer', 7, 'Invoice Reminder Days', 'Days be

-- ============================================================================
-- Token Limits
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('tokens.max_per_request', 'tokens', 'integer', 128000, 'Max Tokens per Request', 'Maximum t
('tokens.max_context_window', 'tokens', 'integer', 200000, 'Max Context Window', 'Maximum co
('tokens.max_output', 'tokens', 'integer', 8192, 'Max Output Tokens', 'Maximum tokens in res
('tokens.streaming_chunk_size', 'tokens', 'integer', 100, 'Streaming Chunk Size', 'Tokens pe

-- ============================================================================
-- Retry Configuration
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('retry.max_attempts', 'retry', 'integer', 3, 'Max Retry Attempts', 'Maximum number of retry
('retry.initial_delay_ms', 'retry', 'integer', 1000, 'Initial Retry Delay', 'Initial delay b
('retry.max_delay_ms', 'retry', 'integer', 30000, 'Max Retry Delay', 'Maximum delay between

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name,
('retry.backoff_multiplier', 'retry', 'decimal', 2.0, 'Backoff Multiplier', 'Multiplier for
('retry.jitter_factor', 'retry', 'decimal', 0.1, 'Jitter Factor', 'Random jitter added to de

-- ============================================================================
-- Cache Settings
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('cache.translation_bundle_ttl', 'cache', 'duration', 300, 'Translation Bundle TTL', 'Cache
('cache.model_list_ttl', 'cache', 'duration', 300, 'Model List TTL', 'Cache duration for AI
('cache.provider_health_ttl', 'cache', 'duration', 60, 'Provider Health TTL', 'Cache duratio
('cache.user_preferences_ttl', 'cache', 'duration', 600, 'User Preferences TTL', 'Cache dura
('cache.config_ttl', 'cache', 'duration', 300, 'Configuration TTL', 'Cache duration for syst

-- ============================================================================
-- Thresholds
-- ============================================================================
```

```
INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('thresholds.health_check.healthy', 'thresholds', 'integer', 2, 'Healthy Threshold', 'Consec
('thresholds.health_check.unhealthy', 'thresholds', 'integer', 3, 'Unhealthy Threshold', 'Co

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name,
('thresholds.confidence.minimum', 'thresholds', 'percentage', 0.70, 'Minimum Confidence', 'M
('thresholds.confidence.high', 'thresholds', 'percentage', 0.90, 'High Confidence', 'Thresho
('thresholds.autoscaling.target_utilization', 'thresholds', 'percentage', 0.70, 'Target Util

-- ============================================================================
-- Volume Discounts
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('discounts.tier_1.threshold', 'discounts', 'integer', 1000000, 'Tier 1 Threshold', 'Token t
('discounts.tier_2.threshold', 'discounts', 'integer', 10000000, 'Tier 2 Threshold', 'Token
('discounts.tier_3.threshold', 'discounts', 'integer', 100000000, 'Tier 3 Threshold', 'Token

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name,
('discounts.tier_1.percentage', 'discounts', 'percentage', 0.05, 'Tier 1 Discount', 'Discoun
('discounts.tier_2.percentage', 'discounts', 'percentage', 0.10, 'Tier 2 Discount', 'Discoun
('discounts.tier_3.percentage', 'discounts', 'percentage', 0.15, 'Tier 3 Discount', 'Discoun

-- ============================================================================
-- Session & Auth
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('session.token_expiry', 'session', 'duration', 3600, 'Token Expiry', 'Access token expirati
('session.refresh_token_expiry', 'session', 'duration', 2592000, 'Refresh Token Expiry', 'Re
('session.max_per_user', 'session', 'integer', 5, 'Max Sessions per User', 'Maximum concurre
('session.invitation_expiry', 'session', 'duration', 604800, 'Invitation Expiry', 'Admin inv

-- ============================================================================
-- Translation System
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('translation.concurrent_limit', 'translation', 'integer', 10, 'Concurrent Translations', 'M
('translation.per_minute_limit', 'translation', 'integer', 100, 'Translations per Minute',
('translation.per_hour_limit', 'translation', 'integer', 1000, 'Translations per Hour', 'Max
('translation.retry_attempts', 'translation', 'integer', 3, 'Translation Retry Attempts', 'M
('translation.retry_delay_ms', 'translation', 'integer', 1000, 'Translation Retry Delay', 'I
('translation.queue_batch_size', 'translation', 'integer', 50, 'Translation Queue Batch', 'M
```

```sql
-- ============================================================================
-- Workflow Proposals
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('workflow.proposal.min_occurrences', 'workflow', 'integer', 5, 'Min Occurrences', 'Minimum
('workflow.proposal.min_unique_users', 'workflow', 'integer', 3, 'Min Unique Users', 'Minimu
('workflow.proposal.min_time_span_hours', 'workflow', 'integer', 24, 'Min Time Span', 'Minin
('workflow.proposal.max_per_day', 'workflow', 'integer', 10, 'Max Proposals per Day', 'Maxin
('workflow.proposal.max_per_week', 'workflow', 'integer', 30, 'Max Proposals per Week', 'Max

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name,
('workflow.proposal.min_impact_score', 'workflow', 'percentage', 0.60, 'Min Impact Score', '
('workflow.proposal.min_confidence', 'workflow', 'percentage', 0.75, 'Min Confidence', 'Mini

-- ============================================================================
-- Notifications
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_json, display_name, de
('notifications.usage_alert_thresholds', 'notifications', 'json', '[50, 75, 90, 100]', 'Usag

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name,
('notifications.alert_cooldown', 'notifications', 'duration', 3600, 'Alert Cooldown', 'Minin
('notifications.digest_frequency', 'notifications', 'duration', 86400, 'Digest Frequency', '

INSERT INTO system_configuration (key, category_id, value_type, value_boolean, display_name,
('notifications.email_enabled', 'notifications', 'boolean', true, 'Email Notifications', 'En
('notifications.slack_enabled', 'notifications', 'boolean', false, 'Slack Notifications', 'S
('notifications.webhook_enabled', 'notifications', 'boolean', false, 'Webhook Notifications'
```

42.4 TYPESCRIPT TYPES

File: `packages/shared/src/config/types.ts`

```typescript
// ============================================================================
// RADIANT v4.8.0 - Configuration Management Types
// ============================================================================

/**
 * Configuration value types
 */
export type ConfigValueType =
  | 'string'
  | 'integer'
```

```
  | 'decimal'
  | 'boolean'
  | 'json'
  | 'duration'
  | 'percentage'
  | 'enum';

/**
 * Configuration category
 */
export interface ConfigCategory {
  id: string;
  name: string;
  description?: string;
  displayOrder: number;
  icon?: string;
}

/**
 * System configuration entry
 */
export interface SystemConfig {
  id: string;
  key: string;
  categoryId: string;
  valueType: ConfigValueType;
  value: string | number | boolean | object;
  displayName: string;
  description?: string;
  unit?: string;
  minValue?: number;
  maxValue?: number;
  enumValues?: string[];
  regexPattern?: string;
  environment: 'all' | 'dev' | 'staging' | 'prod';
  isSensitive: boolean;
  requiresRestart: boolean;
  isDeprecated: boolean;
  deprecatedReplacementKey?: string;
  createdAt: string;
  updatedAt: string;
  updatedBy?: string;
}

/**
 * Tenant configuration override
```

```typescript
 */
export interface TenantConfigOverride {
  id: string;
  tenantId: string;
  configId: string;
  value: string | number | boolean | object;
  validFrom: string;
  validUntil?: string;
  reason?: string;
  createdAt: string;
  updatedAt: string;
  createdBy?: string;
  updatedBy?: string;
}

/**
 * Configuration with resolved value (after tenant override)
 */
export interface ResolvedConfig {
  key: string;
  value: string | number | boolean | object;
  type: ConfigValueType;
  isOverride: boolean;
  displayName?: string;
  unit?: string;
}

/**
 * Configuration audit log entry
 */
export interface ConfigAuditEntry {
  id: string;
  configId?: string;
  tenantOverrideId?: string;
  configKey: string;
  tenantId?: string;
  action: 'created' | 'updated' | 'deleted' | 'override_created' | 'override_updated' | 'ove
  oldValue?: object;
  newValue?: object;
  changedBy?: string;
  changedByEmail?: string;
  reason?: string;
  ipAddress?: string;
  userAgent?: string;
  createdAt: string;
}
```

```typescript
/**
 * Configuration update request
 */
export interface ConfigUpdateRequest {
  value: string | number | boolean | object;
  reason?: string;
}

/**
 * Tenant override request
 */
export interface TenantOverrideRequest {
  value: string | number | boolean | object;
  reason?: string;
  validFrom?: string;
  validUntil?: string;
}

/**
 * Configuration export/import format
 */
export interface ConfigExport {
  version: string;
  exportedAt: string;
  exportedBy: string;
  configs: Array<{
    key: string;
    value: string | number | boolean | object;
  }>;
  tenantOverrides?: Array<{
    tenantId: string;
    key: string;
    value: string | number | boolean | object;
    reason?: string;
  }>;
}
```

File: packages/shared/src/config/constants.ts

```typescript
// ============================================================================
// RADIANT v4.8.0 - Configuration Constants
// ============================================================================

/**
 * Configuration categories
```

```typescript
 */
export const CONFIG_CATEGORIES = {
  RATE_LIMITS: 'rate_limits',
  TIMEOUTS: 'timeouts',
  PRICING: 'pricing',
  TOKENS: 'tokens',
  RETRY: 'retry',
  CACHE: 'cache',
  THRESHOLDS: 'thresholds',
  DISCOUNTS: 'discounts',
  SESSION: 'session',
  TRANSLATION: 'translation',
  WORKFLOW: 'workflow',
  NOTIFICATIONS: 'notifications',
} as const;

/**
 * Common configuration keys
 */
export const CONFIG_KEYS = {
  // Rate Limits
  API_REQUESTS_PER_MINUTE: 'rate_limits.api.requests_per_minute',
  API_REQUESTS_PER_HOUR: 'rate_limits.api.requests_per_hour',
  API_CONCURRENT_CONNECTIONS: 'rate_limits.api.concurrent_connections',
  CHAT_MESSAGES_PER_MINUTE: 'rate_limits.chat.messages_per_minute',

  // Timeouts
  LAMBDA_DEFAULT_TIMEOUT: 'timeouts.lambda.default',
  LAMBDA_CHAT_TIMEOUT: 'timeouts.lambda.chat',
  EXTERNAL_PROVIDER_TIMEOUT: 'timeouts.request.external_provider',
  SESSION_IDLE_TIMEOUT: 'timeouts.session.idle',

  // Pricing
  EXTERNAL_PROVIDER_MARGIN: 'pricing.external_provider_margin',
  SELF_HOSTED_MARGIN: 'pricing.self_hosted_margin',
  MINIMUM_CHARGE: 'pricing.minimum_charge',

  // Tokens
  MAX_TOKENS_PER_REQUEST: 'tokens.max_per_request',
  MAX_CONTEXT_WINDOW: 'tokens.max_context_window',
  MAX_OUTPUT_TOKENS: 'tokens.max_output',

  // Retry
  MAX_RETRY_ATTEMPTS: 'retry.max_attempts',
  INITIAL_RETRY_DELAY: 'retry.initial_delay_ms',
  BACKOFF_MULTIPLIER: 'retry.backoff_multiplier',
```

```
  // Cache
  TRANSLATION_BUNDLE_TTL: 'cache.translation_bundle_ttl',
  MODEL_LIST_TTL: 'cache.model_list_ttl',
  CONFIG_TTL: 'cache.config_ttl',

  // Thresholds
  MIN_CONFIDENCE: 'thresholds.confidence.minimum',
  HIGH_CONFIDENCE: 'thresholds.confidence.high',

  // Translation
  TRANSLATION_CONCURRENT_LIMIT: 'translation.concurrent_limit',
  TRANSLATION_PER_MINUTE_LIMIT: 'translation.per_minute_limit',

  // Workflow
  WORKFLOW_MIN_OCCURRENCES: 'workflow.proposal.min_occurrences',
  WORKFLOW_MIN_UNIQUE_USERS: 'workflow.proposal.min_unique_users',
} as const;

export type ConfigKey = typeof CONFIG_KEYS[keyof typeof CONFIG_KEYS];
```

42.5 CONFIGURATION SERVICE

File: `packages/shared/src/config/ConfigurationService.ts`

```
// =============================================================================
// RADIANT v4.8.0 - Configuration Service
// =============================================================================

import { Pool } from 'pg';
import Redis from 'ioredis';
import {
  SystemConfig,
  ResolvedConfig,
  ConfigValueType,
  ConfigUpdateRequest,
  TenantOverrideRequest,
  ConfigAuditEntry
} from './types';
import { CONFIG_KEYS } from './constants';

export class ConfigurationService {
  private pool: Pool;
  private redis: Redis | null;
  private localCache: Map<string, { value: ResolvedConfig; expiresAt: number }>;
```

```
private defaultTtl: number = 300; // 5 minutes

constructor(pool: Pool, redis?: Redis) {
  this.pool = pool;
  this.redis = redis || null;
  this.localCache = new Map();
}

/**
 * Get a configuration value with tenant override support
 */
async get<T = any>(
  key: string,
  tenantId?: string,
  environment: string = 'prod'
): Promise<T> {
  const cacheKey = this.buildCacheKey(key, tenantId, environment);

  // Check local cache first
  const cached = this.localCache.get(cacheKey);
  if (cached && cached.expiresAt > Date.now()) {
    return cached.value.value as T;
  }

  // Check Redis cache
  if (this.redis) {
    const redisValue = await this.redis.get(cacheKey);
    if (redisValue) {
      const parsed = JSON.parse(redisValue) as ResolvedConfig;
      this.localCache.set(cacheKey, {
        value: parsed,
        expiresAt: Date.now() + this.defaultTtl * 1000
      });
      return parsed.value as T;
    }
  }

  // Fetch from database
  const result = await this.pool.query(
    `SELECT * FROM get_config($1, $2, $3)`,
    [key, tenantId, environment]
  );

  if (!result.rows[0]) {
    throw new Error(`Configuration not found: ${key}`);
  }
```

```
  const config = result.rows[0] as ResolvedConfig;
  const value = this.parseValue(config);

  // Update caches
  const resolvedConfig = { ...config, value };

  if (this.redis) {
    await this.redis.setex(cacheKey, this.defaultTtl, JSON.stringify(resolvedConfig));
  }

  this.localCache.set(cacheKey, {
    value: resolvedConfig,
    expiresAt: Date.now() + this.defaultTtl * 1000
  });

  return value as T;
}

/**
 * Get multiple configuration values
 */
async getMultiple(
  keys: string[],
  tenantId?: string,
  environment: string = 'prod'
): Promise<Record<string, any>> {
  const results: Record<string, any> = {};

  await Promise.all(
    keys.map(async (key) => {
      try {
        results[key] = await this.get(key, tenantId, environment);
      } catch {
        results[key] = undefined;
      }
    })
  );

  return results;
}

/**
 * Get all configurations in a category
 */
async getByCategory(
```

```
  categoryId: string,
  tenantId?: string,
  environment: string = 'prod'
): Promise<ResolvedConfig[]> {
  const result = await this.pool.query(
    `SELECT * FROM get_configs_by_category($1, $2, $3)`,
    [categoryId, tenantId, environment]
  );

  return result.rows;
}

/**
 * Update a global configuration value
 */
async update(
  key: string,
  request: ConfigUpdateRequest,
  updatedBy: string
): Promise<SystemConfig> {
  const config = await this.getConfigByKey(key);

  // Validate value
  this.validateValue(config, request.value);

  // Update based on type
  const updateColumn = this.getValueColumn(config.valueType);

  const result = await this.pool.query(`
    UPDATE system_configuration
    SET ${updateColumn} = $2,
        updated_at = NOW(),
        updated_by = $3
    WHERE key = $1
    RETURNING *
  `, [key, request.value, updatedBy]);

  // Invalidate caches
  await this.invalidateCache(key);

  return result.rows[0];
}

/**
 * Create a tenant-specific override
 */
```

```
async createTenantOverride(
  key: string,
  tenantId: string,
  request: TenantOverrideRequest,
  createdBy: string
): Promise<TenantConfigOverride> {
  const config = await this.getConfigByKey(key);

  // Validate value
  this.validateValue(config, request.value);

  const valueColumn = this.getValueColumn(config.valueType);

  const result = await this.pool.query(`
    INSERT INTO tenant_configuration_overrides (
      tenant_id, config_id, ${valueColumn}, valid_from, valid_until, reason, created_by
    ) VALUES ($1, $2, $3, COALESCE($4, NOW()), $5, $6, $7)
    ON CONFLICT (tenant_id, config_id) DO UPDATE SET
      ${valueColumn} = $3,
      valid_from = COALESCE($4, NOW()),
      valid_until = $5,
      reason = $6,
      updated_by = $7,
      updated_at = NOW()
    RETURNING *
  `, [tenantId, config.id, request.value, request.validFrom, request.validUntil, request.r

  // Invalidate caches
  await this.invalidateCache(key, tenantId);

  return result.rows[0];
}

/**
 * Delete a tenant override (revert to global)
 */
async deleteTenantOverride(
  key: string,
  tenantId: string
): Promise<void> {
  const config = await this.getConfigByKey(key);

  await this.pool.query(`
    DELETE FROM tenant_configuration_overrides
    WHERE config_id = $1 AND tenant_id = $2
  `, [config.id, tenantId]);
```

```
  await this.invalidateCache(key, tenantId);
}

/**
 * Get audit log for a configuration
 */
async getAuditLog(
  key: string,
  options: { limit?: number; offset?: number; tenantId?: string } = {}
): Promise<ConfigAuditEntry[]> {
  const { limit = 50, offset = 0, tenantId } = options;

  let query = `
    SELECT * FROM configuration_audit_log
    WHERE config_key = $1
  `;
  const params: any[] = [key];

  if (tenantId) {
    query += ` AND (tenant_id = $${params.length + 1} OR tenant_id IS NULL)`;
    params.push(tenantId);
  }

  query += ` ORDER BY created_at DESC LIMIT $${params.length + 1} OFFSET $${params.length
  params.push(limit, offset);

  const result = await this.pool.query(query, params);
  return result.rows;
}

/**
 * Export all configurations
 */
async exportConfigs(tenantId?: string): Promise<ConfigExport> {
  const configs = await this.pool.query(`
    SELECT key,
           COALESCE(value_string, value_integer::text, value_decimal::text, value_boolean
    FROM system_configuration
    WHERE is_deprecated = false
  `);

  let tenantOverrides: any[] = [];
  if (tenantId) {
    const overrides = await this.pool.query(`
      SELECT sc.key,
```

```
            COALESCE(tco.value_string, tco.value_integer::text, tco.value_decimal::text,
            tco.reason
      FROM tenant_configuration_overrides tco
      JOIN system_configuration sc ON sc.id = tco.config_id
      WHERE tco.tenant_id = $1
    `, [tenantId]);

    tenantOverrides = overrides.rows.map(r => ({
      tenantId,
      key: r.key,
      value: r.value,
      reason: r.reason,
    }));
  }

  return {
    version: '4.8.0',
    exportedAt: new Date().toISOString(),
    exportedBy: 'system',
    configs: configs.rows,
    tenantOverrides: tenantOverrides.length > 0 ? tenantOverrides : undefined,
  };
}

// Private helpers

private buildCacheKey(key: string, tenantId?: string, environment?: string): string {
  return `config:${environment || 'prod'}:${tenantId || 'global'}:${key}`;
}

private async getConfigByKey(key: string): Promise<SystemConfig> {
  const result = await this.pool.query(
    `SELECT * FROM system_configuration WHERE key = $1`,
    [key]
  );

  if (!result.rows[0]) {
    throw new Error(`Configuration not found: ${key}`);
  }

  return result.rows[0];
}

private getValueColumn(type: ConfigValueType): string {
  switch (type) {
    case 'string':
```

```
    case 'enum':
      return 'value_string';
    case 'integer':
    case 'duration':
      return 'value_integer';
    case 'decimal':
    case 'percentage':
      return 'value_decimal';
    case 'boolean':
      return 'value_boolean';
    case 'json':
      return 'value_json';
    default:
      throw new Error(`Unknown value type: ${type}`);
  }
}

private parseValue(config: ResolvedConfig): any {
  const value = config.value;

  if (typeof value === 'string') {
    try {
      return JSON.parse(value);
    } catch {
      return value;
    }
  }

  return value;
}

private validateValue(config: SystemConfig, value: any): void {
  const numValue = typeof value === 'number' ? value : parseFloat(value);

  if (config.minValue !== undefined && numValue < config.minValue) {
    throw new Error(`Value must be at least ${config.minValue}`);
  }

  if (config.maxValue !== undefined && numValue > config.maxValue) {
    throw new Error(`Value must be at most ${config.maxValue}`);
  }

  if (config.enumValues && !config.enumValues.includes(String(value))) {
    throw new Error(`Value must be one of: ${config.enumValues.join(', ')}`);
  }
```

```
    if (config.regexPattern) {
      const regex = new RegExp(config.regexPattern);
      if (!regex.test(String(value))) {
        throw new Error(`Value does not match required pattern`);
      }
    }
  }

  private async invalidateCache(key: string, tenantId?: string): Promise<void> {
    // Clear local cache
    for (const cacheKey of this.localCache.keys()) {
      if (cacheKey.includes(key)) {
        this.localCache.delete(cacheKey);
      }
    }

    // Clear Redis cache
    if (this.redis) {
      const pattern = tenantId
        ? `config:*:${tenantId}:${key}`
        : `config:*:*:${key}`;

      const keys = await this.redis.keys(pattern);
      if (keys.length > 0) {
        await this.redis.del(...keys);
      }
    }

    // Add to invalidation queue for other instances
    await this.pool.query(`
      INSERT INTO configuration_cache_invalidation (config_key, tenant_id)
      VALUES ($1, $2)
    `, [key, tenantId]);
  }
}

// Singleton instance
let configService: ConfigurationService | null = null;

export function getConfigService(pool: Pool, redis?: Redis): ConfigurationService {
  if (!configService) {
    configService = new ConfigurationService(pool, redis);
  }
  return configService;
}
```

```typescript
/**
 * Helper function to get a config value
 */
export async function getConfig<T = any>(
  key: string,
  tenantId?: string,
  environment?: string
): Promise<T> {
  if (!configService) {
    throw new Error('ConfigurationService not initialized');
  }
  return configService.get<T>(key, tenantId, environment);
}
```

42.6 ADMIN DASHBOARD - CONFIGURATION MANAGEMENT

File: `admin-dashboard/app/configuration/page.tsx`

```typescript
// ============================================================================
// RADIANT v4.8.0 - Configuration Management Dashboard
// ============================================================================

'use client';

import React, { useState, useEffect } from 'react';
import {
  Box,
  Typography,
  Tabs,
  Tab,
  Card,
  CardContent,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  Paper,
  Chip,
  IconButton,
  Button,
  TextField,
```

```
  Dialog,
  DialogTitle,
  DialogContent,
  DialogActions,
  Switch,
  Slider,
  Select,
  MenuItem,
  Alert,
  Tooltip,
  InputAdornment,
  Collapse,
} from '@mui/material';
import {
  Edit,
  History,
  Refresh,
  Settings,
  Speed,
  Timer,
  AttachMoney,
  Token,
  Cached,
  TrendingUp,
  Discount,
  Lock,
  Translate,
  AccountTree,
  Notifications,
  ExpandMore,
  ExpandLess,
  Warning,
  Check,
} from '@mui/icons-material';
import { useTranslation } from '@/hooks/useTranslation';
import { api } from '@/lib/api';

interface ConfigCategory {
  id: string;
  name: string;
  description: string;
  icon: string;
}

interface SystemConfig {
  id: string;
```

```
  key: string;
  categoryId: string;
  valueType: string;
  value: any;
  displayName: string;
  description: string;
  unit: string;
  minValue: number;
  maxValue: number;
  enumValues: string[];
  isSensitive: boolean;
  requiresRestart: boolean;
  isOverride: boolean;
  updatedAt: string;
}

const CATEGORY_ICONS: Record<string, React.ReactNode> = {
  rate_limits: <Speed />,
  timeouts: <Timer />,
  pricing: <AttachMoney />,
  tokens: <Token />,
  retry: <Refresh />,
  cache: <Cached />,
  thresholds: <TrendingUp />,
  discounts: <Discount />,
  session: <Lock />,
  translation: <Translate />,
  workflow: <AccountTree />,
  notifications: <Notifications />,
};

export default function ConfigurationPage() {
  const { t } = useTranslation();
  const [categories, setCategories] = useState<ConfigCategory[]>([]);
  const [activeCategory, setActiveCategory] = useState<string>('rate_limits');
  const [configs, setConfigs] = useState<SystemConfig[]>([]);
  const [editDialog, setEditDialog] = useState<SystemConfig | null>(null);
  const [editValue, setEditValue] = useState<any>(null);
  const [historyDialog, setHistoryDialog] = useState<string | null>(null);
  const [auditLog, setAuditLog] = useState<any[]>([]);
  const [loading, setLoading] = useState(true);
  const [expandedKeys, setExpandedKeys] = useState<Set<string>>(new Set());

  useEffect(() => {
    loadCategories();
  }, []);
```

```
useEffect(() => {
  if (activeCategory) {
    loadConfigs(activeCategory);
  }
}, [activeCategory]);

async function loadCategories() {
  try {
    const res = await api.get('/configuration/categories');
    setCategories(res.data);
    if (res.data.length > 0) {
      setActiveCategory(res.data[0].id);
    }
  } catch (error) {
    console.error('Failed to load categories:', error);
  }
}

async function loadConfigs(categoryId: string) {
  setLoading(true);
  try {
    const res = await api.get(`/configuration/category/${categoryId}`);
    setConfigs(res.data);
  } catch (error) {
    console.error('Failed to load configs:', error);
  } finally {
    setLoading(false);
  }
}

async function handleSave() {
  if (!editDialog) return;
  try {
    await api.put(`/configuration/${editDialog.key}`, {
      value: editValue,
      reason: 'Updated via Admin Dashboard',
    });
    setEditDialog(null);
    loadConfigs(activeCategory);
  } catch (error) {
    console.error('Failed to save config:', error);
  }
}

async function loadAuditLog(key: string) {
```

```
  try {
    const res = await api.get(`/configuration/${key}/audit`);
    setAuditLog(res.data);
    setHistoryDialog(key);
  } catch (error) {
    console.error('Failed to load audit log:', error);
  }
}

function renderValueEditor(config: SystemConfig) {
  switch (config.valueType) {
    case 'boolean':
      return (
        <Switch
          checked={editValue === true || editValue === 'true'}
          onChange={(e) => setEditValue(e.target.checked)}
        />
      );

    case 'integer':
    case 'duration':
      return (
        <Box>
          <Slider
            value={Number(editValue)}
            onChange={(_, v) => setEditValue(v)}
            min={config.minValue || 0}
            max={config.maxValue || 100}
            valueLabelDisplay="auto"
          />
          <TextField
            type="number"
            value={editValue}
            onChange={(e) => setEditValue(Number(e.target.value))}
            InputProps={{
              endAdornment: config.unit && (
                <InputAdornment position="end">{config.unit}</InputAdornment>
              ),
            }}
            fullWidth
            sx={{ mt: 2 }}
          />
        </Box>
      );

    case 'decimal':
```

```
case 'percentage':
  return (
    <Box>
      <Slider
        value={Number(editValue) * (config.valueType === 'percentage' ? 100 : 1)}
        onChange={(_, v) => setEditValue(
          (v as number) / (config.valueType === 'percentage' ? 100 : 1)
        )}
        min={(config.minValue || 0) * (config.valueType === 'percentage' ? 100 : 1)}
        max={(config.maxValue || 1) * (config.valueType === 'percentage' ? 100 : 1)}
        valueLabelDisplay="auto"
        valueLabelFormat={(v) => config.valueType === 'percentage' ? `${v}%` : v}
      />
      <TextField
        type="number"
        value={config.valueType === 'percentage' ? (editValue * 100).toFixed(0) : edit
        onChange={(e) => {
          const val = Number(e.target.value);
          setEditValue(config.valueType === 'percentage' ? val / 100 : val);
        }}
        InputProps={{
          endAdornment: <InputAdornment position="end">%</InputAdornment>,
        }}
        fullWidth
        sx={{ mt: 2 }}
      />
    </Box>
  );

case 'enum':
  return (
    <Select
      value={editValue}
      onChange={(e) => setEditValue(e.target.value)}
      fullWidth
    >
      {config.enumValues?.map((val) => (
        <MenuItem key={val} value={val}>{val}</MenuItem>
      ))}
    </Select>
  );

case 'json':
  return (
    <TextField
      multiline
```

```
            rows={6}
            value={typeof editValue === 'string' ? editValue : JSON.stringify(editValue, nul
            onChange={(e) => {
              try {
                setEditValue(JSON.parse(e.target.value));
              } catch {
                setEditValue(e.target.value);
              }
            }}
            fullWidth
            sx={{ fontFamily: 'monospace' }}
          />
        );

      default:
        return (
          <TextField
            value={editValue}
            onChange={(e) => setEditValue(e.target.value)}
            fullWidth
          />
        );
    }
  }

  function formatValue(config: SystemConfig): string {
    if (config.isSensitive) return '••••••••';

    switch (config.valueType) {
      case 'boolean':
        return config.value ? 'Enabled' : 'Disabled';
      case 'percentage':
        return `${(config.value * 100).toFixed(0)}%`;
      case 'duration':
        if (config.value >= 86400) return `${(config.value / 86400).toFixed(1)} days`;
        if (config.value >= 3600) return `${(config.value / 3600).toFixed(1)} hours`;
        if (config.value >= 60) return `${(config.value / 60).toFixed(0)} min`;
        return `${config.value} sec`;
      case 'json':
        return JSON.stringify(config.value);
      default:
        return `${config.value}${config.unit ? ` ${config.unit}` : ''}`;
    }
  }

  return (
```

```
<Box sx={{ p: 3 }}>
  <Box sx={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center', mb:
    <Typography variant="h4">
      <Settings sx={{ mr: 1, verticalAlign: 'middle' }} />
      {t('admin.configuration.title')}
    </Typography>
    <Button
      variant="outlined"
      startIcon={<Refresh />}
      onClick={() => loadConfigs(activeCategory)}
    >
      {t('admin.buttons.refresh')}
    </Button>
  </Box>

  <Box sx={{ display: 'flex', gap: 3 }}>
    {/* Category Tabs (Vertical) */}
    <Paper sx={{ minWidth: 250 }}>
      <Tabs
        orientation="vertical"
        value={activeCategory}
        onChange={(_, v) => setActiveCategory(v)}
        sx={{ borderRight: 1, borderColor: 'divider' }}
      >
        {categories.map((cat) => (
          <Tab
            key={cat.id}
            value={cat.id}
            label={
              <Box sx={{ display: 'flex', alignItems: 'center', gap: 1 }}>
                {CATEGORY_ICONS[cat.id]}
                <span>{cat.name}</span>
              </Box>
            }
            sx={{ justifyContent: 'flex-start' }}
          />
        ))}
      </Tabs>
    </Paper>

    {/* Configuration Table */}
    <Box sx={{ flex: 1 }}>
      <TableContainer component={Paper}>
        <Table>
          <TableHead>
            <TableRow>
```

```
              <TableCell>{t('admin.configuration.parameter')}</TableCell>
              <TableCell>{t('admin.configuration.value')}</TableCell>
              <TableCell>{t('admin.configuration.status')}</TableCell>
              <TableCell align="right">{t('admin.configuration.actions')}</TableCell>
            </TableRow>
          </TableHead>
          <TableBody>
            {configs.map((config) => (
              <React.Fragment key={config.key}>
                <TableRow hover>
                  <TableCell>
                    <Box>
                      <Typography fontWeight="medium">
                        {config.displayName}
                        {config.requiresRestart && (
                          <Tooltip title="Requires restart">
                            <Warning fontSize="small" color="warning" sx={{ ml: 1 }} />
                          </Tooltip>
                        )}
                      </Typography>
                      <Typography variant="caption" color="text.secondary">
                        {config.key}
                      </Typography>
                    </Box>
                  </TableCell>
                  <TableCell>
                    <Typography fontFamily="monospace">
                      {formatValue(config)}
                    </Typography>
                  </TableCell>
                  <TableCell>
                    {config.isOverride ? (
                      <Chip size="small" color="info" label="Override" />
                    ) : (
                      <Chip size="small" color="default" label="Default" />
                    )}
                  </TableCell>
                  <TableCell align="right">
                    <Tooltip title={t('admin.buttons.edit')}>
                      <IconButton
                        size="small"
                        onClick={() => {
                          setEditDialog(config);
                          setEditValue(config.value);
                        }}
                      >
```

```
                    <Edit fontSize="small" />
                  </IconButton>
                </Tooltip>
                <Tooltip title={t('admin.configuration.history')}>
                  <IconButton
                    size="small"
                    onClick={() => loadAuditLog(config.key)}
                  >
                    <History fontSize="small" />
                  </IconButton>
                </Tooltip>
                <IconButton
                  size="small"
                  onClick={() => {
                    const newSet = new Set(expandedKeys);
                    if (newSet.has(config.key)) {
                      newSet.delete(config.key);
                    } else {
                      newSet.add(config.key);
                    }
                    setExpandedKeys(newSet);
                  }}
                >
                  {expandedKeys.has(config.key) ? <ExpandLess /> : <ExpandMore />}
                </IconButton>
              </TableCell>
            </TableRow>
            <TableRow>
              <TableCell colSpan={4} sx={{ py: 0 }}>
                <Collapse in={expandedKeys.has(config.key)}>
                  <Box sx={{ p: 2, bgcolor: 'grey.50' }}>
                    <Typography variant="body2" color="text.secondary">
                      {config.description}
                    </Typography>
                    {config.minValue !== null && (
                      <Typography variant="caption" display="block">
                        Min: {config.minValue} | Max: {config.maxValue}
                      </Typography>
                    )}
                  </Box>
                </Collapse>
              </TableCell>
            </TableRow>
          </React.Fragment>
        ))}
      </TableBody>
```

```
      </Table>
    </TableContainer>
  </Box>
</Box>

{/* Edit Dialog */}
<Dialog open={!!editDialog} onClose={() => setEditDialog(null)} maxWidth="sm" fullWidt
  <DialogTitle>
    {t('admin.configuration.edit_parameter')}
  </DialogTitle>
  <DialogContent>
    {editDialog && (
      <Box sx={{ pt: 2 }}>
        <Typography variant="subtitle2" gutterBottom>
          {editDialog.displayName}
        </Typography>
        <Typography variant="body2" color="text.secondary" gutterBottom>
          {editDialog.description}
        </Typography>
        <Box sx={{ mt: 3 }}>
          {renderValueEditor(editDialog)}
        </Box>
        {editDialog.requiresRestart && (
          <Alert severity="warning" sx={{ mt: 2 }}>
            {t('admin.configuration.requires_restart_warning')}
          </Alert>
        )}
      </Box>
    )}
  </DialogContent>
  <DialogActions>
    <Button onClick={() => setEditDialog(null)}>
      {t('admin.buttons.cancel')}
    </Button>
    <Button onClick={handleSave} variant="contained" color="primary">
      {t('admin.buttons.save')}
    </Button>
  </DialogActions>
</Dialog>

{/* History Dialog */}
<Dialog open={!!historyDialog} onClose={() => setHistoryDialog(null)} maxWidth="md" fu
  <DialogTitle>
    {t('admin.configuration.change_history')}: {historyDialog}
  </DialogTitle>
  <DialogContent>
```

```
            <TableContainer>
              <Table size="small">
                <TableHead>
                  <TableRow>
                    <TableCell>{t('admin.configuration.date')}</TableCell>
                    <TableCell>{t('admin.configuration.action')}</TableCell>
                    <TableCell>{t('admin.configuration.old_value')}</TableCell>
                    <TableCell>{t('admin.configuration.new_value')}</TableCell>
                    <TableCell>{t('admin.configuration.changed_by')}</TableCell>
                  </TableRow>
                </TableHead>
                <TableBody>
                  {auditLog.map((entry) => (
                    <TableRow key={entry.id}>
                      <TableCell>
                        {new Date(entry.createdAt).toLocaleString()}
                      </TableCell>
                      <TableCell>
                        <Chip size="small" label={entry.action} />
                      </TableCell>
                      <TableCell>
                        <code>{JSON.stringify(entry.oldValue?.value)}</code>
                      </TableCell>
                      <TableCell>
                        <code>{JSON.stringify(entry.newValue?.value)}</code>
                      </TableCell>
                      <TableCell>{entry.changedByEmail || 'System'}</TableCell>
                    </TableRow>
                  ))}
                </TableBody>
              </Table>
            </TableContainer>
          </DialogContent>
          <DialogActions>
            <Button onClick={() => setHistoryDialog(null)}>
              {t('admin.buttons.close')}
            </Button>
          </DialogActions>
        </Dialog>
      </Box>
    );
  }
```

42.7 API LAMBDA

File: `lambda/configuration/api.ts`

```
// ============================================================================
// RADIANT v4.8.0 - Configuration API Lambda
// ============================================================================

import { APIGatewayProxyHandler, APIGatewayProxyResult } from 'aws-lambda';
import { Pool } from 'pg';
import { extractAuthContext, requireRoles } from '../shared/auth';
import { ConfigurationService } from '@radiant/shared/config';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false },
});

const configService = new ConfigurationService(pool);

export const handler: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxyResult>
  const { httpMethod, path, pathParameters, body } = event;

  try {
    const auth = extractAuthContext(event);
    requireRoles(auth, ['admin', 'super_admin']);

    // GET /configuration/categories
    if (httpMethod === 'GET' && path === '/configuration/categories') {
      return await getCategories();
    }

    // GET /configuration/category/:id
    if (httpMethod === 'GET' && path.match(/\/configuration\/category\/[a-z_]+$/)) {
      return await getByCategory(pathParameters?.id!, auth.tenantId);
    }

    // GET /configuration/:key
    if (httpMethod === 'GET' && path.match(/\/configuration\/[a-z_.]+$/)) {
      return await getConfig(pathParameters?.key!, auth.tenantId);
    }

    // PUT /configuration/:key
    if (httpMethod === 'PUT' && path.match(/\/configuration\/[a-z_.]+$/)) {
      requireRoles(auth, ['super_admin']);
      return await updateConfig(pathParameters?.key!, JSON.parse(body || '{}'), auth.userId)
```

```
    }

    // GET /configuration/:key/audit
    if (httpMethod === 'GET' && path.match(/\/configuration\/[a-z_.]+\/audit$/)) {
      return await getAuditLog(pathParameters?.key!, auth.tenantId);
    }

    // POST /configuration/:key/tenant-override
    if (httpMethod === 'POST' && path.match(/\/configuration\/[a-z_.]+\/tenant-override$/))
      requireRoles(auth, ['super_admin']);
      const { tenantId, ...rest } = JSON.parse(body || '{}');
      return await createTenantOverride(pathParameters?.key!, tenantId, rest, auth.userId);
    }

    // DELETE /configuration/:key/tenant-override/:tenantId
    if (httpMethod === 'DELETE' && path.match(/\/configuration\/[a-z_.]+\/tenant-override\/
      requireRoles(auth, ['super_admin']);
      return await deleteTenantOverride(pathParameters?.key!, pathParameters?.tenantId!);
    }

    // POST /configuration/export
    if (httpMethod === 'POST' && path === '/configuration/export') {
      return await exportConfigs(auth.tenantId);
    }

    return { statusCode: 404, body: JSON.stringify({ error: 'Not found' }) };
  } catch (error) {
    console.error('Configuration API error:', error);
    return {
      statusCode: error.statusCode || 500,
      body: JSON.stringify({ error: error.message }),
    };
  }
};

async function getCategories(): Promise<APIGatewayProxyResult> {
  const result = await pool.query(`
    SELECT id, name, description, display_order, icon
    FROM configuration_categories
    ORDER BY display_order
  `);

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows),
  };
```

```
}

async function getByCategory(categoryId: string, tenantId: string): Promise<APIGatewayProxyR
  const configs = await configService.getByCategory(categoryId, tenantId);

  return {
    statusCode: 200,
    body: JSON.stringify(configs),
  };
}

async function getConfig(key: string, tenantId: string): Promise<APIGatewayProxyResult> {
  const value = await configService.get(key, tenantId);

  return {
    statusCode: 200,
    body: JSON.stringify({ key, value }),
  };
}

async function updateConfig(key: string, data: any, userId: string): Promise<APIGatewayProxy
  const config = await configService.update(key, data, userId);

  return {
    statusCode: 200,
    body: JSON.stringify(config),
  };
}

async function getAuditLog(key: string, tenantId: string): Promise<APIGatewayProxyResult> {
  const log = await configService.getAuditLog(key, { tenantId, limit: 100 });

  return {
    statusCode: 200,
    body: JSON.stringify(log),
  };
}

async function createTenantOverride(
  key: string,
  tenantId: string,
  data: any,
  userId: string
): Promise<APIGatewayProxyResult> {
  const override = await configService.createTenantOverride(key, tenantId, data, userId);
```

```
  return {
    statusCode: 201,
    body: JSON.stringify(override),
  };
}

async function deleteTenantOverride(key: string, tenantId: string): Promise<APIGatewayProxyR
  await configService.deleteTenantOverride(key, tenantId);

  return {
    statusCode: 204,
    body: '',
  };
}

async function exportConfigs(tenantId: string): Promise<APIGatewayProxyResult> {
  const exportData = await configService.exportConfigs(tenantId);

  return {
    statusCode: 200,
    headers: {
      'Content-Type': 'application/json',
      'Content-Disposition': `attachment; filename="radiant-config-${new Date().toISOString(
    },
    body: JSON.stringify(exportData, null, 2),
  };
}
```

42.8 USAGE EXAMPLE - UPDATING CODE TO USE CONFIGURABLE VALUES

Before (Hardcoded):

```
// OLD CODE - Hardcoded values
const DEFAULT_MARGIN = 0.40;
const MAX_RETRY_ATTEMPTS = 3;
const SESSION_TIMEOUT = 1800;

async function calculateCost(providerCost: number): Promise<number> {
  return providerCost * (1 + DEFAULT_MARGIN);  // ← HARDCODED!
}

async function retryRequest(fn: () => Promise<any>): Promise<any> {
  for (let i = 0; i < MAX_RETRY_ATTEMPTS; i++) {  // ← HARDCODED!
```

```
    try {
      return await fn();
    } catch (error) {
      if (i === MAX_RETRY_ATTEMPTS - 1) throw error;
      await sleep(1000 * Math.pow(2, i));
    }
  }
}
```

After (Database-Driven):

```
// NEW CODE - Database-driven with ConfigurationService
import { getConfig, CONFIG_KEYS } from '@radiant/shared/config';

async function calculateCost(providerCost: number, tenantId: string): Promise<number> {
  const margin = await getConfig<number>(
    CONFIG_KEYS.EXTERNAL_PROVIDER_MARGIN,
    tenantId
  );
  return providerCost * (1 + margin);  //  CONFIGURABLE!
}

async function retryRequest(fn: () => Promise<any>, tenantId: string): Promise<any> {
  const [maxAttempts, initialDelay, backoffMultiplier] = await Promise.all([
    getConfig<number>(CONFIG_KEYS.MAX_RETRY_ATTEMPTS, tenantId),
    getConfig<number>(CONFIG_KEYS.INITIAL_RETRY_DELAY, tenantId),
    getConfig<number>(CONFIG_KEYS.BACKOFF_MULTIPLIER, tenantId),
  ]);

  for (let i = 0; i < maxAttempts; i++) {  //  CONFIGURABLE!
    try {
      return await fn();
    } catch (error) {
      if (i === maxAttempts - 1) throw error;
      await sleep(initialDelay * Math.pow(backoffMultiplier, i));
    }
  }
}
```

42.9 LOCALIZATION STRINGS FOR CONFIGURATION UI

Add to migration 041b_seed_localization.sql:

```
-- Configuration Management Strings
INSERT INTO localization_registry (key, default_text, category, context, source_app) VALUES
('admin.configuration.title', 'Configuration Management', 'features.admin', 'Page title', 'a
('admin.configuration.parameter', 'Parameter', 'features.admin', 'Table header', 'admin'),
('admin.configuration.value', 'Value', 'features.admin', 'Table header', 'admin'),
('admin.configuration.status', 'Status', 'features.admin', 'Table header', 'admin'),
('admin.configuration.actions', 'Actions', 'features.admin', 'Table header', 'admin'),
('admin.configuration.history', 'History', 'features.admin', 'Button tooltip', 'admin'),
('admin.configuration.edit_parameter', 'Edit Parameter', 'features.admin', 'Dialog title', '
('admin.configuration.change_history', 'Change History', 'features.admin', 'Dialog title', '
('admin.configuration.date', 'Date', 'features.admin', 'Table header', 'admin'),
('admin.configuration.action', 'Action', 'features.admin', 'Table header', 'admin'),
('admin.configuration.old_value', 'Old Value', 'features.admin', 'Table header', 'admin'),
('admin.configuration.new_value', 'New Value', 'features.admin', 'Table header', 'admin'),
('admin.configuration.changed_by', 'Changed By', 'features.admin', 'Table header', 'admin');
('admin.configuration.requires_restart_warning', 'This change requires a service restart to
```

IMPLEMENTATION VERIFICATION CHECKLIST

Complete v4.8.0 Verification

Section 42: Dynamic Configuration Management System

- ☐ Migration 042_configuration_management.sql applied successfully
- ☐ Migration 042b_seed_configuration.sql applied with all parameters
- ☐ configuration_categories table created with 12 categories
- ☐ system_configuration table created with type validation
- ☐ tenant_configuration_overrides table created with RLS
- ☐ configuration_audit_log table created
- ☐ configuration_cache_invalidation table created
- ☐ get_config() function working with tenant override support
- ☐ get_configs_by_category() function returning configs
- ☐ Audit log triggers capturing all changes
- ☐ Cache invalidation triggers working

- ☐ ConfigurationService TypeScript class implemented
- ☐ getConfig() helper function working with caching
- ☐ Redis caching layer integrated
- ☐ Configuration API Lambda deployed
- ☐ Admin /configuration page accessible
- ☐ Category tabs displaying all 12 categories
- ☐ Edit dialog with type-appropriate editors
- ☐ History dialog showing audit log
- ☐ Tenant override creation working
- ☐ Configuration export/import working
- ☐ All hardcoded values replaced with getConfig() calls
- ☐ Localization strings added for configuration UI

Section 41: Complete Internationalization System (from v4.7.0)

- ☐ All previous v4.7.0 checklist items verified

Section 40: Application-Level Data Isolation (from v4.6.0)

- ☐ All previous v4.6.0 checklist items verified

Integration Verification

- ☐ All Lambda functions using ConfigurationService
- ☐ No hardcoded rate limits remaining
- ☐ No hardcoded timeouts remaining
- ☐ No hardcoded pricing margins remaining
- ☐ No hardcoded token limits remaining
- ☐ No hardcoded retry configurations remaining
- ☐ Configuration changes propagate without deployment
- ☐ Per-tenant overrides working correctly
- ☐ Audit trail complete for all changes

SECTION-43-BILLING-CREDITS

SECTION 43: BILLING & CREDITS SYSTEM (v4.13.0)

Version: 4.13.0 | Adds 7-tier subscriptions and prepaid credits system

---

43.1 BILLING SYSTEM OVERVIEW

```
                    RADIANT v4.13.0 - BILLING SYSTEM


       7-TIER SUBSCRIPTION MODEL
       (Similar to OpenAI/ChatGPT)
```

- FREE - Trial ($0, 0.5 credits)
- INDIVIDUAL - Personal ($29/mo)
- FAMILY - Household ($49/mo, 5 users)
- TEAM - Small biz ($25/user, 2-25)
- BUSINESS - Mid-market ($45/user)
- ENTERPRISE - Large orgs (Custom)
- ENTERPRISE PLUS - Compliance incl.

```
       PREPAID CREDITS SYSTEM
       (Similar to Windsurf Pro)
```

- $10 = 1 Credit (any quantity)
- Volume discounts (5% to 25% off)
- Bonus credits for bulk purchases
- Credit pools for families/teams
- Auto-purchase when balance low
- Credits never expire

```
       COMPLIANCE ADD-ON ($25/user/mo)
       Available for TEAM+ tiers
```

- HIPAA + BAA
- SOC 2 Type II
- GDPR/CCPA compliance
- Customer-managed encryption keys
- Enhanced audit logging
- Data residency options

43.2 SUBSCRIPTION TIERS

packages/shared/src/billing/tiers.ts

```typescript
/**
 * RADIANT Subscription Tiers
 * 7-tier model from FREE to ENTERPRISE PLUS
 *
 * @version 4.13.0
 * @module @radiant/shared/billing
 */

// ============================================================================
// TIER DEFINITIONS
// ============================================================================

export type SubscriptionTierId =
  | 'FREE'
  | 'INDIVIDUAL'
  | 'FAMILY'
  | 'TEAM'
  | 'BUSINESS'
  | 'ENTERPRISE'
  | 'ENTERPRISE_PLUS';

export interface SubscriptionTier {
  id: SubscriptionTierId;
  displayName: string;  // Fallback - use localizationKey at runtime
  description: string;  // Fallback - use localizationKey at runtime
  localizationKey: string;  // e.g., 'billing.tiers.free' for i18n lookup

  // Pricing
  pricing: {
    monthly: number | null;           // null = contact sales
```

```
    annual: number | null;
    perUser: boolean;
    minUsers?: number;
    maxUsers?: number;
    currency: string;
    contactSales?: boolean;
  };

  // Credits
  includedCreditsPerUser: number;     // Monthly credits per user/seat

  // Rate limits
  rateLimits: {
    requestsPerMinute: number;
    tokensPerMinute: number;
    concurrentRequests: number;
  };

  // Features
  features: TierFeatures;

  // Add-ons available
  availableAddOns: string[];

  // Display
  isPublic: boolean;
  sortOrder: number;
  badge?: string;
}

export interface TierFeatures {
  modelAccess: 'limited' | 'full' | 'full_plus_beta' | 'all';
  maxModelsPerMonth: number | null;
  maxRequestsPerDay: number | null;
  maxTokensPerRequest: number;
  priorityQueue: boolean;
  apiAccess: boolean;
  exportFormats: string[];
  supportLevel: 'community' | 'email' | 'priority_email' | 'priority' | 'dedicated';
  dataRetention: string;
  watermarkedOutputs: boolean;
  conversationHistory: boolean;
  customInstructions: boolean;

  // Sharing features
  sharedCreditPool: boolean;
```

```
  teamWorkspaces: boolean;
  sharedConversations: boolean;

  // Admin features
  teamAdminDashboard: boolean;
  usageAnalytics: boolean;
  roleBasedAccess: boolean;
  inviteManagement: boolean;

  // Enterprise features
  ssoIntegration: boolean;
  scimProvisioning: boolean;
  auditLogs: boolean;
  dataExport: boolean;
  customBranding: boolean;
  dedicatedAccountManager: boolean;
  apiRateLimitCustomization: boolean;
  webhooks: boolean;
  ipWhitelisting: boolean;

  // Family features
  parentalControls?: boolean;
  familyAdminDashboard?: boolean;
  perMemberUsageLimits?: boolean;

  // Enterprise Plus features
  slaGuarantee?: boolean;
  uptimeGuarantee?: string;
  customModelFineTuning?: boolean;
  dedicatedInfrastructure?: boolean;
  multiRegionDeployment?: boolean;
  onPremiseDeployment?: boolean;
}

// ============================================================================
// TIER REGISTRY
// ============================================================================

export const SUBSCRIPTION_TIERS: Record<SubscriptionTierId, SubscriptionTier> = {
  FREE: {
    id: 'FREE',
    displayName: 'Free Trial',  // Fallback
    description: 'Try RADIANT with limited features',  // Fallback
    localizationKey: 'billing.tiers.free',
    pricing: { monthly: 0, annual: 0, perUser: false, currency: 'USD' },
    includedCreditsPerUser: 0.5,
```

```
  rateLimits: { requestsPerMinute: 10, tokensPerMinute: 20_000, concurrentRequests: 1 },
  features: {
    modelAccess: 'limited', maxModelsPerMonth: 5, maxRequestsPerDay: 50,
    maxTokensPerRequest: 4_000, priorityQueue: false, apiAccess: false,
    exportFormats: ['txt'], supportLevel: 'community', dataRetention: '7_days',
    watermarkedOutputs: true, conversationHistory: false, customInstructions: false,
    sharedCreditPool: false, teamWorkspaces: false, sharedConversations: false,
    teamAdminDashboard: false, usageAnalytics: false, roleBasedAccess: false,
    inviteManagement: false, ssoIntegration: false, scimProvisioning: false,
    auditLogs: false, dataExport: false, customBranding: false,
    dedicatedAccountManager: false, apiRateLimitCustomization: false,
    webhooks: false, ipWhitelisting: false,
  },
  availableAddOns: [],
  isPublic: true,
  sortOrder: 0,
},

INDIVIDUAL: {
  id: 'INDIVIDUAL',
  displayName: 'Individual',  // Fallback
  description: 'Full-featured personal plan',  // Fallback
  localizationKey: 'billing.tiers.individual',
  pricing: { monthly: 29, annual: 290, perUser: false, currency: 'USD' },
  includedCreditsPerUser: 3,
  rateLimits: { requestsPerMinute: 60, tokensPerMinute: 100_000, concurrentRequests: 5 },
  features: {
    modelAccess: 'full', maxModelsPerMonth: null, maxRequestsPerDay: 1_000,
    maxTokensPerRequest: 128_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx'], supportLevel: 'email',
    dataRetention: '90_days', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: false, teamWorkspaces: false,
    sharedConversations: false, teamAdminDashboard: false, usageAnalytics: false,
    roleBasedAccess: false, inviteManagement: false, ssoIntegration: false,
    scimProvisioning: false, auditLogs: false, dataExport: false,
    customBranding: false, dedicatedAccountManager: false,
    apiRateLimitCustomization: false, webhooks: false, ipWhitelisting: false,
  },
  availableAddOns: ['API_POWER_PACK'],
  isPublic: true,
  sortOrder: 1,
},

FAMILY: {
  id: 'FAMILY',
  displayName: 'Family',  // Fallback
```

```
  description: 'Share AI across your household',  // Fallback
  localizationKey: 'billing.tiers.family',
  id: 'FAMILY',
  displayName: 'Family',
  description: 'Share AI across your household',
  pricing: { monthly: 49, annual: 490, perUser: false, maxUsers: 5, currency: 'USD' },
  includedCreditsPerUser: 6,
  rateLimits: { requestsPerMinute: 120, tokensPerMinute: 200_000, concurrentRequests: 10 }
  features: {
    modelAccess: 'full', maxModelsPerMonth: null, maxRequestsPerDay: 2_000,
    maxTokensPerRequest: 128_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx'], supportLevel: 'email',
    dataRetention: '90_days', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: true, teamWorkspaces: false,
    sharedConversations: false, teamAdminDashboard: false, usageAnalytics: false,
    roleBasedAccess: false, inviteManagement: true, ssoIntegration: false,
    scimProvisioning: false, auditLogs: false, dataExport: false,
    customBranding: false, dedicatedAccountManager: false,
    apiRateLimitCustomization: false, webhooks: false, ipWhitelisting: false,
    parentalControls: true, familyAdminDashboard: true, perMemberUsageLimits: true,
  },
  availableAddOns: ['API_POWER_PACK'],
  isPublic: true,
  sortOrder: 2,
},

TEAM: {
  id: 'TEAM',
  displayName: 'Team',
  description: 'Collaborate with your small team',
  pricing: { monthly: 25, annual: 250, perUser: true, minUsers: 2, maxUsers: 25, currency
  includedCreditsPerUser: 3,
  rateLimits: { requestsPerMinute: 300, tokensPerMinute: 500_000, concurrentRequests: 20 }
  features: {
    modelAccess: 'full', maxModelsPerMonth: null, maxRequestsPerDay: null,
    maxTokensPerRequest: 200_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx', 'html'], supportLevel: 'priority_email',
    dataRetention: '1_year', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: true, teamWorkspaces: true,
    sharedConversations: true, teamAdminDashboard: true, usageAnalytics: true,
    roleBasedAccess: true, inviteManagement: true, ssoIntegration: false,
    scimProvisioning: false, auditLogs: false, dataExport: false,
    customBranding: false, dedicatedAccountManager: false,
    apiRateLimitCustomization: false, webhooks: false, ipWhitelisting: false,
  },
  availableAddOns: ['COMPLIANCE_ADDON', 'PRIORITY_SUPPORT'],
```

```
  isPublic: true,
  sortOrder: 3,
  badge: 'Most Popular',
},

BUSINESS: {
  id: 'BUSINESS',
  displayName: 'Business',
  description: 'Enterprise features for growing companies',
  pricing: { monthly: 45, annual: 450, perUser: true, minUsers: 5, maxUsers: 150, currency
  includedCreditsPerUser: 5,
  rateLimits: { requestsPerMinute: 1_000, tokensPerMinute: 2_000_000, concurrentRequests:
  features: {
    modelAccess: 'full_plus_beta', maxModelsPerMonth: null, maxRequestsPerDay: null,
    maxTokensPerRequest: 500_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx', 'html', 'json'], supportLevel: 'priority',
    dataRetention: '2_years', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: true, teamWorkspaces: true,
    sharedConversations: true, teamAdminDashboard: true, usageAnalytics: true,
    roleBasedAccess: true, inviteManagement: true, ssoIntegration: true,
    scimProvisioning: true, auditLogs: true, dataExport: true,
    customBranding: true, dedicatedAccountManager: false,
    apiRateLimitCustomization: true, webhooks: true, ipWhitelisting: true,
  },
  availableAddOns: ['COMPLIANCE_ADDON', 'PRIORITY_SUPPORT'],
  isPublic: true,
  sortOrder: 4,
},

ENTERPRISE: {
  id: 'ENTERPRISE',
  displayName: 'Enterprise',
  description: 'Full-scale enterprise deployment',
  pricing: { monthly: null, annual: null, perUser: true, minUsers: 50, currency: 'USD', co
  includedCreditsPerUser: 10,
  rateLimits: { requestsPerMinute: 5_000, tokensPerMinute: 10_000_000, concurrentRequests
  features: {
    modelAccess: 'all', maxModelsPerMonth: null, maxRequestsPerDay: null,
    maxTokensPerRequest: 1_000_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx', 'html', 'json', 'csv'],
    supportLevel: 'dedicated', dataRetention: 'custom', watermarkedOutputs: false,
    conversationHistory: true, customInstructions: true, sharedCreditPool: true,
    teamWorkspaces: true, sharedConversations: true, teamAdminDashboard: true,
    usageAnalytics: true, roleBasedAccess: true, inviteManagement: true,
    ssoIntegration: true, scimProvisioning: true, auditLogs: true, dataExport: true,
    customBranding: true, dedicatedAccountManager: true,
```

```
      apiRateLimitCustomization: true, webhooks: true, ipWhitelisting: true,
      slaGuarantee: true, uptimeGuarantee: '99.9%', customModelFineTuning: true,
      multiRegionDeployment: true,
    },
    availableAddOns: ['COMPLIANCE_ADDON'],
    isPublic: true,
    sortOrder: 5,
  },

  ENTERPRISE_PLUS: {
    id: 'ENTERPRISE_PLUS',
    displayName: 'Enterprise Plus',
    description: 'Maximum security and compliance for regulated industries',
    pricing: { monthly: null, annual: null, perUser: true, minUsers: 100, currency: 'USD', c
    includedCreditsPerUser: 15,
    rateLimits: { requestsPerMinute: 20_000, tokensPerMinute: 50_000_000, concurrentRequests
    features: {
      modelAccess: 'all', maxModelsPerMonth: null, maxRequestsPerDay: null,
      maxTokensPerRequest: 2_000_000, priorityQueue: true, apiAccess: true,
      exportFormats: ['txt', 'md', 'pdf', 'docx', 'html', 'json', 'csv', 'xml'],
      supportLevel: 'dedicated', dataRetention: 'custom', watermarkedOutputs: false,
      conversationHistory: true, customInstructions: true, sharedCreditPool: true,
      teamWorkspaces: true, sharedConversations: true, teamAdminDashboard: true,
      usageAnalytics: true, roleBasedAccess: true, inviteManagement: true,
      ssoIntegration: true, scimProvisioning: true, auditLogs: true, dataExport: true,
      customBranding: true, dedicatedAccountManager: true,
      apiRateLimitCustomization: true, webhooks: true, ipWhitelisting: true,
      slaGuarantee: true, uptimeGuarantee: '99.99%', customModelFineTuning: true,
      dedicatedInfrastructure: true, multiRegionDeployment: true, onPremiseDeployment: true,
    },
    availableAddOns: [], // Compliance included
    isPublic: true,
    sortOrder: 6,
    badge: 'Full Compliance Included',
  },
};

// ============================================================================
// HELPER FUNCTIONS
// ============================================================================

export function getTier(tierId: SubscriptionTierId): SubscriptionTier {
  return SUBSCRIPTION_TIERS[tierId];
}

export function canUpgrade(fromTier: SubscriptionTierId, toTier: SubscriptionTierId): boolea
```

```
  return SUBSCRIPTION_TIERS[toTier].sortOrder > SUBSCRIPTION_TIERS[fromTier].sortOrder;
}

export function getAnnualSavings(tier: SubscriptionTier): number {
  if (!tier.pricing.monthly || !tier.pricing.annual) return 0;
  return (tier.pricing.monthly * 12) - tier.pricing.annual;
}
```

43.3 CREDITS SYSTEM

packages/shared/src/billing/credits.ts

```
/**
 * RADIANT Credits System
 * Prepaid AI usage currency
 *
 * @version 4.13.0
 * @module @radiant/shared/billing
 */

// ============================================================================
// CREDIT TYPES
// ============================================================================

export interface CreditPool {
  id: string;
  type: CreditPoolType;
  ownerId: string;
  organizationId?: string;
  tenantId: string;

  balance: {
    available: number;
    reserved: number;
    total: number;
  };

  monthlyIncluded: {
    total: number;
    used: number;
    remaining: number;
    resetsAt: string;
  };

  purchased: {
```

```
    total: number;
    remaining: number;
    lastPurchaseAt?: string;
  };

  bonus: {
    total: number;
    remaining: number;
  };

  members: CreditPoolMember[];
  settings: CreditPoolSettings;

  createdAt: string;
  updatedAt: string;
}

export type CreditPoolType = 'individual' | 'family' | 'team' | 'organization' | 'enterprise

export interface CreditPoolMember {
  userId: string;
  email: string;
  displayName: string;
  role: 'owner' | 'admin' | 'member' | 'restricted';

  limits?: {
    dailyCredits?: number;
    monthlyCredits?: number;
    maxCostPerRequest?: number;
  };

  permissions: CreditPoolPermissions;
  status: 'active' | 'invited' | 'suspended';
  joinedAt: string;
  lastActiveAt?: string;

  usage: {
    today: number;
    thisWeek: number;
    thisMonth: number;
    allTime: number;
  };
}

export interface CreditPoolPermissions {
  canPurchaseCredits: boolean;
```

```
  canViewPoolBalance: boolean;
  canViewOtherUsage: boolean;
  canInviteMembers: boolean;
  canRemoveMembers: boolean;
  canModifySettings: boolean;
}

export interface CreditPoolSettings {
  autoPurchase: {
    enabled: boolean;
    thresholdCredits: number;
    purchaseAmount: number;
    maxMonthlyAutoPurchase?: number;
    paymentMethodId?: string;
  };

  notifications: {
    lowBalanceThreshold: number;
    lowBalanceRecipients: string[];
    usageSummaryFrequency: 'daily' | 'weekly' | 'monthly';
    unusualUsageAlerts: boolean;
  };

  accessControl: {
    requireApprovalAbove?: number;
    blockedModels?: string[];
    allowedModels?: string[];
  };

  familySettings?: {
    parentalControlsEnabled: boolean;
    contentFiltering: 'strict' | 'moderate' | 'off';
    underageMembers: string[];
  };
}

// ============================================================================
// CREDIT TRANSACTIONS
// ============================================================================

export type CreditTransactionType =
  | 'purchase' | 'subscription_grant' | 'bonus_grant' | 'consumption'
  | 'refund' | 'adjustment' | 'transfer_in' | 'transfer_out' | 'expiration';

export interface CreditTransaction {
  id: string;
```

```
  poolId: string;
  userId?: string;
  transactionType: CreditTransactionType;

  amount: number;
  balanceAfter: number;

  sourceType?: 'included' | 'purchased' | 'bonus';
  sourceId?: string;

  modelId?: string;
  requestId?: string;
  inputTokens?: number;
  outputTokens?: number;

  paymentIntentId?: string;
  paymentAmountCents?: number;
  paymentCurrency?: string;

  description?: string;
  metadata?: Record<string, any>;

  createdAt: string;
}

// ============================================================================
// CREDIT PRICING
// ============================================================================

export const CREDIT_PRICING = {
  basePrice: 10.00, // $10 = 1 Credit

  volumeDiscounts: [
    { minCredits: 1, discount: 0, bonus: 0 },
    { minCredits: 5, discount: 0, bonus: 0 },
    { minCredits: 10, discount: 0.05, bonus: 0.05 },   // 5% off, 5% bonus
    { minCredits: 25, discount: 0.10, bonus: 0.10 },   // 10% off, 10% bonus
    { minCredits: 50, discount: 0.15, bonus: 0.15 },   // 15% off, 15% bonus
    { minCredits: 100, discount: 0.20, bonus: 0.20 },  // 20% off, 20% bonus
  ],

  consumption: {
    text: {
      inputTokensPer1Credit: 1_000_000,
      outputTokensPer1Credit: 250_000,
    },
```

```
    image: {
      standardGenerationCredits: 0.02,
      hdGenerationCredits: 0.04,
    },
    audio: {
      transcriptionMinuteCredits: 0.01,
      ttsCharacterCredits: 0.00001,
    },
  },
};

export function calculatePurchasePrice(credits: number): {
  basePrice: number;
  discount: number;
  bonusCredits: number;
  finalPrice: number;
  totalCredits: number;
} {
  const tier = [...CREDIT_PRICING.volumeDiscounts]
    .reverse()
    .find(t => credits >= t.minCredits) || CREDIT_PRICING.volumeDiscounts[0];

  const basePrice = credits * CREDIT_PRICING.basePrice;
  const discount = basePrice * tier.discount;
  const bonusCredits = credits * tier.bonus;

  return {
    basePrice,
    discount,
    bonusCredits,
    finalPrice: basePrice - discount,
    totalCredits: credits + bonusCredits,
  };
}
```

43.4 DATABASE SCHEMA

packages/infrastructure/migrations/043_billing_system.sql

```
-- ============================================================================
-- RADIANT v4.13.0 - Billing & Credits Schema
-- Migration: 043_billing_system.sql
-- ============================================================================

-- Subscription Tiers
```

```sql
CREATE TABLE subscription_tiers (
    id VARCHAR(50) PRIMARY KEY,
    display_name VARCHAR(100) NOT NULL,
    description TEXT,

    price_monthly DECIMAL(10,2),
    price_annual DECIMAL(10,2),
    price_per_user BOOLEAN DEFAULT FALSE,
    min_users INTEGER DEFAULT 1,
    max_users INTEGER,
    currency VARCHAR(3) DEFAULT 'USD',

    included_credits_per_user DECIMAL(10,2) NOT NULL DEFAULT 0,

    requests_per_minute INTEGER NOT NULL,
    tokens_per_minute BIGINT NOT NULL,
    concurrent_requests INTEGER NOT NULL,

    features JSONB NOT NULL DEFAULT '{}',
    available_add_ons TEXT[] DEFAULT '{}',

    is_active BOOLEAN DEFAULT TRUE,
    is_public BOOLEAN DEFAULT TRUE,
    sort_order INTEGER DEFAULT 0,
    badge VARCHAR(50),

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Subscription Add-Ons
CREATE TABLE subscription_add_ons (
    id VARCHAR(50) PRIMARY KEY,
    display_name VARCHAR(100) NOT NULL,
    description TEXT,

    price_monthly_cents INTEGER NOT NULL,
    price_annual_cents INTEGER,
    price_per_user BOOLEAN DEFAULT FALSE,

    available_for_tiers TEXT[] NOT NULL,
    included_in_tiers TEXT[] DEFAULT '{}',

    features JSONB NOT NULL DEFAULT '{}',

    is_active BOOLEAN DEFAULT TRUE,
```

```
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Credit Pools
CREATE TABLE credit_pools (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_type VARCHAR(20) NOT NULL CHECK (pool_type IN ('individual', 'family', 'team', 'org
    owner_user_id UUID NOT NULL REFERENCES users(id),
    organization_id UUID REFERENCES organizations(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),

    balance_available DECIMAL(12,4) NOT NULL DEFAULT 0,
    balance_reserved DECIMAL(12,4) NOT NULL DEFAULT 0,

    monthly_included_total DECIMAL(12,4) NOT NULL DEFAULT 0,
    monthly_included_used DECIMAL(12,4) NOT NULL DEFAULT 0,
    monthly_resets_at TIMESTAMPTZ,

    purchased_total DECIMAL(12,4) NOT NULL DEFAULT 0,
    purchased_remaining DECIMAL(12,4) NOT NULL DEFAULT 0,
    last_purchase_at TIMESTAMPTZ,

    bonus_total DECIMAL(12,4) NOT NULL DEFAULT 0,
    bonus_remaining DECIMAL(12,4) NOT NULL DEFAULT 0,

    settings JSONB NOT NULL DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_pools_owner ON credit_pools(owner_user_id);
CREATE INDEX idx_credit_pools_org ON credit_pools(organization_id);
CREATE INDEX idx_credit_pools_tenant ON credit_pools(tenant_id);

-- Credit Pool Members
CREATE TABLE credit_pool_members (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),

    role VARCHAR(20) NOT NULL DEFAULT 'member' CHECK (role IN ('owner', 'admin', 'member',
    permissions JSONB NOT NULL DEFAULT '{}',

    daily_credit_limit DECIMAL(10,4),
```

```
    monthly_credit_limit DECIMAL(10,4),
    max_cost_per_request DECIMAL(10,4),

    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'invited', 'sus
    invited_by UUID REFERENCES users(id),
    invited_at TIMESTAMPTZ,
    joined_at TIMESTAMPTZ,
    suspended_at TIMESTAMPTZ,
    suspended_reason TEXT,

    usage_today DECIMAL(12,4) NOT NULL DEFAULT 0,
    usage_this_week DECIMAL(12,4) NOT NULL DEFAULT 0,
    usage_this_month DECIMAL(12,4) NOT NULL DEFAULT 0,
    usage_all_time DECIMAL(12,4) NOT NULL DEFAULT 0,
    last_usage_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(pool_id, user_id)
);

CREATE INDEX idx_pool_members_user ON credit_pool_members(user_id);
CREATE INDEX idx_pool_members_pool ON credit_pool_members(pool_id);

-- Credit Transactions
CREATE TABLE credit_transactions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id),
    user_id UUID REFERENCES users(id),

    transaction_type VARCHAR(30) NOT NULL CHECK (transaction_type IN (
        'purchase', 'subscription_grant', 'bonus_grant', 'consumption',
        'refund', 'adjustment', 'transfer_in', 'transfer_out', 'expiration'
    )),

    amount DECIMAL(12,4) NOT NULL,
    balance_after DECIMAL(12,4) NOT NULL,

    source_type VARCHAR(30),
    source_id VARCHAR(100),

    model_id VARCHAR(100),
    request_id UUID,
    input_tokens INTEGER,
    output_tokens INTEGER,
```

```
    payment_intent_id VARCHAR(100),
    payment_amount_cents INTEGER,
    payment_currency VARCHAR(3),

    description TEXT,
    metadata JSONB DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_transactions_pool ON credit_transactions(pool_id);
CREATE INDEX idx_credit_transactions_user ON credit_transactions(user_id);
CREATE INDEX idx_credit_transactions_type ON credit_transactions(transaction_type);
CREATE INDEX idx_credit_transactions_created ON credit_transactions(created_at);

-- Credit Purchases
CREATE TABLE credit_purchases (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id),
    user_id UUID NOT NULL REFERENCES users(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),

    credits_amount DECIMAL(10,4) NOT NULL,
    bonus_credits DECIMAL(10,4) NOT NULL DEFAULT 0,
    total_credits DECIMAL(10,4) NOT NULL,

    payment_amount_cents INTEGER NOT NULL,
    payment_currency VARCHAR(3) NOT NULL DEFAULT 'USD',
    payment_method VARCHAR(50),

    stripe_payment_intent_id VARCHAR(100),
    stripe_charge_id VARCHAR(100),
    stripe_invoice_id VARCHAR(100),
    stripe_receipt_url TEXT,

    status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (status IN (
        'pending', 'processing', 'completed', 'failed', 'refunded', 'partially_refunded'
    )),
    completed_at TIMESTAMPTZ,
    failed_at TIMESTAMPTZ,
    failure_reason TEXT,

    is_auto_purchase BOOLEAN DEFAULT FALSE,
    auto_purchase_trigger VARCHAR(50),
```

```
    metadata JSONB DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_purchases_pool ON credit_purchases(pool_id);
CREATE INDEX idx_credit_purchases_user ON credit_purchases(user_id);
CREATE INDEX idx_credit_purchases_status ON credit_purchases(status);
CREATE INDEX idx_credit_purchases_stripe ON credit_purchases(stripe_payment_intent_id);

-- Subscriptions
CREATE TABLE subscriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID NOT NULL REFERENCES users(id),
    organization_id UUID REFERENCES organizations(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    credit_pool_id UUID REFERENCES credit_pools(id),

    tier_id VARCHAR(50) NOT NULL REFERENCES subscription_tiers(id),
    billing_cycle VARCHAR(10) NOT NULL CHECK (billing_cycle IN ('monthly', 'annual')),

    seat_count INTEGER NOT NULL DEFAULT 1,
    add_ons JSONB NOT NULL DEFAULT '[]',

    price_per_unit_cents INTEGER NOT NULL,
    total_price_cents INTEGER NOT NULL,
    currency VARCHAR(3) NOT NULL DEFAULT 'USD',

    stripe_subscription_id VARCHAR(100),
    stripe_customer_id VARCHAR(100) NOT NULL,
    stripe_price_id VARCHAR(100),

    current_period_start TIMESTAMPTZ NOT NULL,
    current_period_end TIMESTAMPTZ NOT NULL,
    trial_start TIMESTAMPTZ,
    trial_end TIMESTAMPTZ,

    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN (
        'trialing', 'active', 'past_due', 'canceled', 'unpaid', 'paused'
    )),
    cancel_at_period_end BOOLEAN DEFAULT FALSE,
    canceled_at TIMESTAMPTZ,
    cancellation_reason TEXT,
```

```
    metadata JSONB DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_subscriptions_user ON subscriptions(user_id);
CREATE INDEX idx_subscriptions_org ON subscriptions(organization_id);
CREATE INDEX idx_subscriptions_tenant ON subscriptions(tenant_id);
CREATE INDEX idx_subscriptions_stripe ON subscriptions(stripe_subscription_id);
CREATE INDEX idx_subscriptions_status ON subscriptions(status);

-- Auto-Purchase Settings
CREATE TABLE auto_purchase_settings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id) UNIQUE,

    enabled BOOLEAN NOT NULL DEFAULT FALSE,
    threshold_credits DECIMAL(10,4) NOT NULL DEFAULT 1.0,
    purchase_credits DECIMAL(10,4) NOT NULL DEFAULT 10.0,
    max_monthly_auto_purchase DECIMAL(10,4),

    stripe_payment_method_id VARCHAR(100),

    auto_purchases_this_month INTEGER NOT NULL DEFAULT 0,
    auto_purchase_spend_this_month DECIMAL(10,2) NOT NULL DEFAULT 0,
    last_auto_purchase_at TIMESTAMPTZ,
    month_reset_at TIMESTAMPTZ,

    notify_on_auto_purchase BOOLEAN DEFAULT TRUE,
    notification_emails TEXT[],

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Credit Usage (for analytics)
CREATE TABLE credit_usage (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    pool_id UUID NOT NULL REFERENCES credit_pools(id),
    user_id UUID NOT NULL REFERENCES users(id),
    transaction_id UUID REFERENCES credit_transactions(id),

    request_id UUID NOT NULL,
    model_id VARCHAR(100) NOT NULL,
```

```
    provider_id VARCHAR(50) NOT NULL,

    input_tokens INTEGER NOT NULL DEFAULT 0,
    output_tokens INTEGER NOT NULL DEFAULT 0,
    cached_tokens INTEGER NOT NULL DEFAULT 0,

    credits_consumed DECIMAL(12,6) NOT NULL,
    credit_rate_multiplier DECIMAL(6,4) NOT NULL DEFAULT 1.0,

    credits_from_included DECIMAL(12,6) NOT NULL DEFAULT 0,
    credits_from_purchased DECIMAL(12,6) NOT NULL DEFAULT 0,
    credits_from_bonus DECIMAL(12,6) NOT NULL DEFAULT 0,

    request_type VARCHAR(50),
    latency_ms INTEGER,

    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_usage_pool ON credit_usage(pool_id);
CREATE INDEX idx_credit_usage_user ON credit_usage(user_id);
CREATE INDEX idx_credit_usage_model ON credit_usage(model_id);
CREATE INDEX idx_credit_usage_created ON credit_usage(created_at);

-- Billing Events
CREATE TABLE billing_events (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    event_type VARCHAR(50) NOT NULL,
    stripe_event_id VARCHAR(100),

    pool_id UUID REFERENCES credit_pools(id),
    user_id UUID REFERENCES users(id),
    subscription_id UUID REFERENCES subscriptions(id),
    purchase_id UUID REFERENCES credit_purchases(id),

    data JSONB NOT NULL DEFAULT '{}',

    processed BOOLEAN DEFAULT FALSE,
    processed_at TIMESTAMPTZ,
    error TEXT,
    retry_count INTEGER DEFAULT 0,

    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_billing_events_type ON billing_events(event_type);
```

```sql
CREATE INDEX idx_billing_events_processed ON billing_events(processed) WHERE processed = FAI
CREATE INDEX idx_billing_events_stripe ON billing_events(stripe_event_id);

-- Row Level Security
ALTER TABLE credit_pools ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_pool_members ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_transactions ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_purchases ENABLE ROW LEVEL SECURITY;
ALTER TABLE subscriptions ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_usage ENABLE ROW LEVEL SECURITY;

CREATE POLICY credit_pools_tenant_isolation ON credit_pools
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY credit_pool_members_access ON credit_pool_members FOR SELECT
    USING (
        user_id = current_setting('app.current_user_id')::UUID OR
        pool_id IN (
            SELECT pool_id FROM credit_pool_members
            WHERE user_id = current_setting('app.current_user_id')::UUID
            AND role IN ('owner', 'admin')
        )
    );
```

43.5 CREDITS SERVICE

packages/functions/src/services/credits.ts

```typescript
/**
 * Credits Service
 * Manages credit consumption, purchases, and balances
 *
 * @version 4.13.0
 */

import { Pool } from 'pg';
import Stripe from 'stripe';
import { CREDIT_PRICING, calculatePurchasePrice } from '@radiant/shared/billing/credits';

export class CreditsService {
  constructor(
    private pool: Pool,
    private stripe: Stripe
  ) {}
```

```
async getBalance(userId: string): Promise<{
  available: number;
  reserved: number;
  includedRemaining: number;
  purchasedRemaining: number;
  bonusRemaining: number;
}> {
  const result = await this.pool.query(`
    SELECT
      cp.balance_available,
      cp.balance_reserved,
      cp.monthly_included_total - cp.monthly_included_used as included_remaining,
      cp.purchased_remaining,
      cp.bonus_remaining
    FROM credit_pools cp
    JOIN credit_pool_members cpm ON cpm.pool_id = cp.id
    WHERE cpm.user_id = $1 AND cpm.status = 'active'
  `, [userId]);

  if (result.rows.length === 0) {
    return { available: 0, reserved: 0, includedRemaining: 0, purchasedRemaining: 0, bonus
  }

  const row = result.rows[0];
  return {
    available: parseFloat(row.balance_available),
    reserved: parseFloat(row.balance_reserved),
    includedRemaining: parseFloat(row.included_remaining),
    purchasedRemaining: parseFloat(row.purchased_remaining),
    bonusRemaining: parseFloat(row.bonus_remaining),
  };
}

async reserveCredits(userId: string, amount: number): Promise<boolean> {
  const result = await this.pool.query(`
    UPDATE credit_pools cp
    SET
      balance_available = balance_available - $2,
      balance_reserved = balance_reserved + $2,
      updated_at = NOW()
    FROM credit_pool_members cpm
    WHERE cpm.pool_id = cp.id
      AND cpm.user_id = $1
      AND cpm.status = 'active'
      AND cp.balance_available >= $2
    RETURNING cp.id
```

```
    `, [userId, amount]);

    return result.rowCount > 0;
  }

  async consumeCredits(
    userId: string,
    modelId: string,
    inputTokens: number,
    outputTokens: number,
    requestId: string
  ): Promise<{ consumed: number; remaining: number }> {
    const creditCost = this.calculateCreditCost(modelId, inputTokens, outputTokens);

    const client = await this.pool.connect();
    try {
      await client.query('BEGIN');

      // Consume from reserved first, then available
      const consumeResult = await client.query(`
        UPDATE credit_pools cp
        SET
          balance_reserved = GREATEST(0, balance_reserved - $2),
          balance_available = CASE
            WHEN balance_reserved >= $2 THEN balance_available
            ELSE balance_available - ($2 - balance_reserved)
          END,
          updated_at = NOW()
        FROM credit_pool_members cpm
        WHERE cpm.pool_id = cp.id
          AND cpm.user_id = $1
          AND cpm.status = 'active'
        RETURNING cp.id, cp.balance_available
      `, [userId, creditCost]);

      if (consumeResult.rows.length === 0) {
        throw new Error('No active credit pool found');
      }

      // Record transaction
      await client.query(`
        INSERT INTO credit_transactions (
          pool_id, user_id, transaction_type, amount, balance_after,
          model_id, request_id, input_tokens, output_tokens
        ) VALUES ($1, $2, 'consumption', $3, $4, $5, $6, $7, $8)
      `, [
```

```
        consumeResult.rows[0].id, userId, -creditCost,
        consumeResult.rows[0].balance_available, modelId, requestId,
        inputTokens, outputTokens
      ]);

      // Record usage for analytics
      await client.query(`
        INSERT INTO credit_usage (
          pool_id, user_id, request_id, model_id, provider_id,
          input_tokens, output_tokens, credits_consumed
        ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8)
      `, [
        consumeResult.rows[0].id, userId, requestId, modelId, 'auto',
        inputTokens, outputTokens, creditCost
      ]);

      // Update member usage stats
      await client.query(`
        UPDATE credit_pool_members
        SET
          usage_today = usage_today + $2,
          usage_this_month = usage_this_month + $2,
          usage_all_time = usage_all_time + $2,
          last_usage_at = NOW(),
          updated_at = NOW()
        WHERE user_id = $1
      `, [userId, creditCost]);

      await client.query('COMMIT');

      return {
        consumed: creditCost,
        remaining: parseFloat(consumeResult.rows[0].balance_available),
      };
    } catch (error) {
      await client.query('ROLLBACK');
      throw error;
    } finally {
      client.release();
    }
  }

  async purchaseCredits(
    userId: string,
    poolId: string,
    credits: number,
```

```
  paymentMethodId: string
): Promise<{ purchaseId: string; totalCredits: number }> {
  const pricing = calculatePurchasePrice(credits);

  // Create Stripe payment intent
  const paymentIntent = await this.stripe.paymentIntents.create({
    amount: Math.round(pricing.finalPrice * 100),
    currency: 'usd',
    payment_method: paymentMethodId,
    confirm: true,
    metadata: {
      poolId,
      userId,
      credits: credits.toString(),
      bonusCredits: pricing.bonusCredits.toString(),
    },
  });

  // Record purchase
  const result = await this.pool.query(`
    INSERT INTO credit_purchases (
      pool_id, user_id, tenant_id,
      credits_amount, bonus_credits, total_credits,
      payment_amount_cents, stripe_payment_intent_id,
      status, completed_at
    )
    SELECT
      $1, $2, cp.tenant_id,
      $3, $4, $5, $6, $7, 'completed', NOW()
    FROM credit_pools cp WHERE cp.id = $1
    RETURNING id
  `, [
    poolId, userId,
    credits, pricing.bonusCredits, pricing.totalCredits,
    Math.round(pricing.finalPrice * 100), paymentIntent.id
  ]);

  // Add credits to pool
  await this.pool.query(`
    UPDATE credit_pools
    SET
      purchased_total = purchased_total + $2,
      purchased_remaining = purchased_remaining + $2,
      bonus_total = bonus_total + $3,
      bonus_remaining = bonus_remaining + $3,
      balance_available = balance_available + $2 + $3,
```

```
        last_purchase_at = NOW(),
        updated_at = NOW()
      WHERE id = $1
    `, [poolId, credits, pricing.bonusCredits]);

    return {
      purchaseId: result.rows[0].id,
      totalCredits: pricing.totalCredits,
    };
  }

  private calculateCreditCost(modelId: string, inputTokens: number, outputTokens: number): n
    const inputCredits = inputTokens / CREDIT_PRICING.consumption.text.inputTokensPer1Credit
    const outputCredits = outputTokens / CREDIT_PRICING.consumption.text.outputTokensPer1Cre
    return inputCredits + outputCredits;
  }
}
```

SECTION-44-STORAGE-BILLING

SECTION 44: STORAGE BILLING SYSTEM (v4.14.0)

Version: 4.14.0 | Tiered storage billing for S3 and database usage

---

44.1 STORAGE BILLING OVERVIEW

Tier	S3 ($/GB/mo$)|DB(/GB/mo)	Backup ($/GB/mo)	Included	
FREE	$0	$0	N/A	1GB S3, 500MB DB
INDIVIDUAL	$0.10	$0.15	Included	10GB S3, 2GB DB
FAMILY	$0.08	$0.12	Included	25GB S3, 5GB DB
TEAM	$0.06	$0.10	Included	100GB S3, 20GB DB
BUSINESS	$0.04	$0.08	Included	500GB S3, 100GB DB

Tier	S3 (/GB/mo)|DB(/GB/mo)	Backup ($/GB/mo)	Included	
ENTERPRISE	Custom	Custom	Custom	Unlimited

44.2 DATABASE SCHEMA

packages/infrastructure/migrations/044_storage_billing.sql

```
-- ============================================================================
-- RADIANT v4.14.0 - Storage Billing Schema
-- ============================================================================

-- Storage Usage Tracking
CREATE TABLE storage_usage (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID REFERENCES users(id),
    app_id UUID REFERENCES applications(id),

    storage_type VARCHAR(20) NOT NULL CHECK (storage_type IN ('s3', 'database', 'backup', 'e

    bytes_used BIGINT NOT NULL DEFAULT 0,
    bytes_quota BIGINT,

    period_start TIMESTAMPTZ NOT NULL,
    period_end TIMESTAMPTZ NOT NULL,

    price_per_gb_cents INTEGER NOT NULL,
    total_cost_cents INTEGER NOT NULL DEFAULT 0,

    is_over_quota BOOLEAN DEFAULT FALSE,
    quota_warning_sent BOOLEAN DEFAULT FALSE,
    quota_exceeded_sent BOOLEAN DEFAULT FALSE,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_storage_usage_tenant ON storage_usage(tenant_id);
CREATE INDEX idx_storage_usage_type ON storage_usage(storage_type);
CREATE INDEX idx_storage_usage_period ON storage_usage(period_start, period_end);

-- Storage Pricing Configuration
CREATE TABLE storage_pricing (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tier_id VARCHAR(50) NOT NULL REFERENCES subscription_tiers(id),
    storage_type VARCHAR(20) NOT NULL,

    price_per_gb_cents INTEGER NOT NULL,
    included_gb DECIMAL(10,2) NOT NULL DEFAULT 0,
    max_gb DECIMAL(10,2),
    overage_price_per_gb_cents INTEGER,

    is_active BOOLEAN DEFAULT TRUE,
    effective_from TIMESTAMPTZ DEFAULT NOW(),
    effective_until TIMESTAMPTZ,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tier_id, storage_type, effective_from)
);

-- Storage Events
CREATE TABLE storage_events (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID REFERENCES users(id),

    event_type VARCHAR(30) NOT NULL CHECK (event_type IN (
        'upload', 'delete', 'archive', 'restore', 'expire', 'quota_warning', 'quota_exceeded
    )),

    storage_type VARCHAR(20) NOT NULL,
    bytes_delta BIGINT NOT NULL,

    resource_id VARCHAR(255),
    resource_type VARCHAR(50),
    resource_path TEXT,

    metadata JSONB DEFAULT '{}',
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_storage_events_tenant ON storage_events(tenant_id);
CREATE INDEX idx_storage_events_type ON storage_events(event_type);

-- Enable RLS
ALTER TABLE storage_usage ENABLE ROW LEVEL SECURITY;
ALTER TABLE storage_events ENABLE ROW LEVEL SECURITY;
```

```
CREATE POLICY storage_usage_tenant ON storage_usage
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY storage_events_tenant ON storage_events
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
```

44.3 STORAGE SERVICE

packages/functions/src/services/storage-billing.ts

```
/**
 * Storage Billing Service
 * @version 4.14.0
 */

import { Pool } from 'pg';
import { S3Client, ListObjectsV2Command } from '@aws-sdk/client-s3';

export class StorageBillingService {
  constructor(private pool: Pool, private s3Client: S3Client) {}

  async getStorageUsage(tenantId: string): Promise<{
    storageType: string;
    bytesUsed: number;
    bytesQuota: number | null;
    pricePerGbCents: number;
    includedGb: number;
    totalCostCents: number;
  }[]> {
    const result = await this.pool.query(`
      SELECT
        su.storage_type,
        su.bytes_used,
        su.bytes_quota,
        sp.price_per_gb_cents,
        sp.included_gb,
        CASE
          WHEN su.bytes_used <= sp.included_gb * 1073741824 THEN 0
          ELSE CEIL((su.bytes_used - sp.included_gb * 1073741824) / 1073741824.0) * sp.price
        END as total_cost_cents
      FROM storage_usage su
      JOIN subscriptions s ON s.tenant_id = su.tenant_id AND s.status = 'active'
      JOIN storage_pricing sp ON sp.tier_id = s.tier_id AND sp.storage_type = su.storage_typ
      WHERE su.tenant_id = $1 AND sp.is_active = TRUE AND su.period_end > NOW()
    `, [tenantId]);
```

```
  return result.rows;
}

async recordStorageEvent(
  tenantId: string,
  userId: string | null,
  eventType: string,
  storageType: string,
  bytesDelta: number,
  resourceId?: string
): Promise<void> {
  await this.pool.query(`
    INSERT INTO storage_events (tenant_id, user_id, event_type, storage_type, bytes_delta,
    VALUES ($1, $2, $3, $4, $5, $6)
  `, [tenantId, userId, eventType, storageType, bytesDelta, resourceId]);

  await this.updateStorageUsage(tenantId, storageType, bytesDelta);
}

private async updateStorageUsage(tenantId: string, storageType: string, bytesDelta: numbe
  await this.pool.query(`
    INSERT INTO storage_usage (tenant_id, storage_type, bytes_used, period_start, period_e
    SELECT $1, $2, GREATEST(0, $3), date_trunc('month', NOW()), date_trunc('month', NOW()
      COALESCE((SELECT price_per_gb_cents FROM storage_pricing sp
        JOIN subscriptions s ON s.tier_id = sp.tier_id AND s.tenant_id = $1
        WHERE sp.storage_type = $2 AND sp.is_active = TRUE LIMIT 1), 10)
    ON CONFLICT (tenant_id, storage_type, period_start, period_end)
    DO UPDATE SET bytes_used = GREATEST(0, storage_usage.bytes_used + $3), updated_at = NO
  `, [tenantId, storageType, bytesDelta]);
}

async calculateS3Usage(tenantId: string): Promise<number> {
  let totalBytes = 0;
  let continuationToken: string | undefined;

  do {
    const response = await this.s3Client.send(new ListObjectsV2Command({
      Bucket: process.env.S3_BUCKET_NAME,
      Prefix: `tenants/${tenantId}/`,
      ContinuationToken: continuationToken,
    }));

    if (response.Contents) {
      totalBytes += response.Contents.reduce((sum, obj) => sum + (obj.Size || 0), 0);
    }
```

```
      continuationToken = response.NextContinuationToken;
    } while (continuationToken);

    return totalBytes;
  }
}
```

SECTION-45-VERSIONED-SUBSCRIPTIONS

SECTION 45: VERSIONED SUBSCRIPTIONS & GRANDFATHERING (v4.15.0)

Version: 4.15.0 | Preserves original pricing/features when plans change

---

45.1 GRANDFATHERING PRINCIPLES

- **Existing subscribers keep original terms** when plans change
- **Price increases don't affect** current subscribers
- **Migration offers with incentives** encourage voluntary upgrades
- **Complete audit trail** of all plan changes

45.2 DATABASE SCHEMA

packages/infrastructure/migrations/045_versioned_subscriptions.sql

```
-- ============================================================================
-- RADIANT v4.15.0 - Versioned Subscriptions & Grandfathering
-- ============================================================================

-- Subscription Plan Versions
CREATE TABLE subscription_plan_versions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tier_id VARCHAR(50) NOT NULL REFERENCES subscription_tiers(id),
```

```
    version_number INTEGER NOT NULL,

    display_name VARCHAR(100) NOT NULL,
    description TEXT,

    price_monthly_cents INTEGER,
    price_annual_cents INTEGER,
    price_per_user BOOLEAN DEFAULT FALSE,

    included_credits_per_user DECIMAL(10,2) NOT NULL,
    features JSONB NOT NULL,
    rate_limits JSONB NOT NULL,

    effective_from TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    effective_until TIMESTAMPTZ,

    change_reason TEXT,
    changed_by UUID REFERENCES administrators(id),

    created_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tier_id, version_number)
);

CREATE INDEX idx_plan_versions_tier ON subscription_plan_versions(tier_id);
CREATE INDEX idx_plan_versions_effective ON subscription_plan_versions(effective_from, effe

-- Grandfathered Subscriptions
CREATE TABLE grandfathered_subscriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    subscription_id UUID NOT NULL REFERENCES subscriptions(id),
    plan_version_id UUID NOT NULL REFERENCES subscription_plan_versions(id),

    locked_price_monthly_cents INTEGER,
    locked_price_annual_cents INTEGER,
    locked_features JSONB NOT NULL,
    locked_rate_limits JSONB NOT NULL,
    locked_credits_per_user DECIMAL(10,2) NOT NULL,

    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'opted_out', 'm

    migration_offered BOOLEAN DEFAULT FALSE,
    migration_offer_date TIMESTAMPTZ,
    migration_incentive JSONB,
    migration_response VARCHAR(20),
    migration_response_date TIMESTAMPTZ,
```

```
    grandfathered_at TIMESTAMPTZ DEFAULT NOW(),
    grandfathered_reason TEXT,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_grandfathered_subscription ON grandfathered_subscriptions(subscription_id);
CREATE INDEX idx_grandfathered_status ON grandfathered_subscriptions(status);

-- Plan Change Audit
CREATE TABLE plan_change_audit (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tier_id VARCHAR(50) NOT NULL REFERENCES subscription_tiers(id),

    old_version_id UUID REFERENCES subscription_plan_versions(id),
    new_version_id UUID NOT NULL REFERENCES subscription_plan_versions(id),

    change_type VARCHAR(30) NOT NULL CHECK (change_type IN (
        'price_increase', 'price_decrease', 'feature_add', 'feature_remove',
        'limit_increase', 'limit_decrease', 'credit_change', 'terms_change'
    )),

    change_summary TEXT NOT NULL,
    affected_subscribers INTEGER NOT NULL DEFAULT 0,
    grandfathered_count INTEGER NOT NULL DEFAULT 0,

    changed_by UUID REFERENCES administrators(id),
    approved_by UUID REFERENCES administrators(id),

    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Function: Get effective plan for subscription
CREATE OR REPLACE FUNCTION get_effective_plan(p_subscription_id UUID)
RETURNS TABLE (
    tier_id VARCHAR(50),
    version_number INTEGER,
    price_monthly_cents INTEGER,
    price_annual_cents INTEGER,
    features JSONB,
    rate_limits JSONB,
    credits_per_user DECIMAL(10,2),
    is_grandfathered BOOLEAN
) AS $$
```

```
BEGIN
    RETURN QUERY
    SELECT
        s.tier_id,
        COALESCE(pv.version_number, 0),
        COALESCE(gs.locked_price_monthly_cents, st.price_monthly::INTEGER * 100),
        COALESCE(gs.locked_price_annual_cents, st.price_annual::INTEGER * 100),
        COALESCE(gs.locked_features, st.features),
        COALESCE(gs.locked_rate_limits, jsonb_build_object(
            'requestsPerMinute', st.requests_per_minute,
            'tokensPerMinute', st.tokens_per_minute,
            'concurrentRequests', st.concurrent_requests
        )),
        COALESCE(gs.locked_credits_per_user, st.included_credits_per_user),
        (gs.id IS NOT NULL) as is_grandfathered
    FROM subscriptions s
    JOIN subscription_tiers st ON st.id = s.tier_id
    LEFT JOIN grandfathered_subscriptions gs ON gs.subscription_id = s.id AND gs.status = 'a
    LEFT JOIN subscription_plan_versions pv ON pv.id = gs.plan_version_id
    WHERE s.id = p_subscription_id;
END;
$$ LANGUAGE plpgsql;
```

45.3 GRANDFATHERING SERVICE

packages/functions/src/services/grandfathering.ts

```
/**
 * Grandfathering Service
 * @version 4.15.0
 */

import { Pool } from 'pg';

export class GrandfatheringService {
  constructor(private pool: Pool) {}

  async createPlanVersion(
    tierId: string,
    changeType: string,
    changeSummary: string,
    changedBy: string,
    approvedBy: string
  ): Promise<{ versionId: string; grandfatheredCount: number }> {
    const client = await this.pool.connect();
```

```
try {
  await client.query('BEGIN');

  // Get current tier and latest version
  const tierResult = await client.query(`SELECT * FROM subscription_tiers WHERE id = $1
  const tier = tierResult.rows[0];

  const versionResult = await client.query(`
    SELECT COALESCE(MAX(version_number), 0) + 1 as next_version
    FROM subscription_plan_versions WHERE tier_id = $1
  `, [tierId]);
  const nextVersion = versionResult.rows[0].next_version;

  // Close previous version
  await client.query(`
    UPDATE subscription_plan_versions SET effective_until = NOW()
    WHERE tier_id = $1 AND effective_until IS NULL
  `, [tierId]);

  // Create new version
  const newVersionResult = await client.query(`
    INSERT INTO subscription_plan_versions (
      tier_id, version_number, display_name, description,
      price_monthly_cents, price_annual_cents, price_per_user,
      included_credits_per_user, features, rate_limits, change_reason, changed_by
    ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12)
    RETURNING id
  `, [
    tierId, nextVersion, tier.display_name, tier.description,
    tier.price_monthly ? tier.price_monthly * 100 : null,
    tier.price_annual ? tier.price_annual * 100 : null,
    tier.price_per_user, tier.included_credits_per_user, tier.features,
    JSON.stringify({ requestsPerMinute: tier.requests_per_minute, tokensPerMinute: tier
    changeSummary, changedBy
  ]);

  const newVersionId = newVersionResult.rows[0].id;

  // Get previous version for grandfathering
  const prevVersionResult = await client.query(`
    SELECT id FROM subscription_plan_versions WHERE tier_id = $1 AND version_number = $2
  `, [tierId, nextVersion]);

  let grandfatheredCount = 0;
```

```
    if (prevVersionResult.rows.length > 0) {
      const prevVersionId = prevVersionResult.rows[0].id;

      // Grandfather all active subscriptions
      const grandfatherResult = await client.query(`
        INSERT INTO grandfathered_subscriptions (
          subscription_id, plan_version_id, locked_price_monthly_cents, locked_price_annua
          locked_features, locked_rate_limits, locked_credits_per_user, grandfathered_reas
        )
        SELECT s.id, $1, pv.price_monthly_cents, pv.price_annual_cents,
          pv.features, pv.rate_limits, pv.included_credits_per_user, $2
        FROM subscriptions s
        JOIN subscription_plan_versions pv ON pv.id = $1
        WHERE s.tier_id = $3 AND s.status IN ('active', 'trialing')
          AND NOT EXISTS (SELECT 1 FROM grandfathered_subscriptions gs WHERE gs.subscripti
      `, [prevVersionId, changeSummary, tierId]);

      grandfatheredCount = grandfatherResult.rowCount || 0;
    }

    // Record audit
    await client.query(`
      INSERT INTO plan_change_audit (tier_id, old_version_id, new_version_id, change_type,
      VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)
    `, [tierId, prevVersionResult.rows[0]?.id, newVersionId, changeType, changeSummary, gr

    await client.query('COMMIT');
    return { versionId: newVersionId, grandfatheredCount };
  } catch (error) {
    await client.query('ROLLBACK');
    throw error;
  } finally {
    client.release();
  }
}

async getEffectivePlan(subscriptionId: string): Promise<{
  grandfathered: boolean;
  priceMonthly: number | null;
  features: Record<string, any>;
  rateLimits: Record<string, number>;
  creditsPerUser: number;
}> {
  const result = await this.pool.query(`SELECT * FROM get_effective_plan($1)`, [subscripti
  const row = result.rows[0];
```

```
    return {
      grandfathered: row.is_grandfathered,
      priceMonthly: row.price_monthly_cents,
      features: row.features,
      rateLimits: row.rate_limits,
      creditsPerUser: row.credits_per_user,
    };
  }

  async offerMigration(subscriptionId: string, incentive: { bonusCredits?: number; discountP
    await this.pool.query(`
      UPDATE grandfathered_subscriptions
      SET migration_offered = TRUE, migration_offer_date = NOW(), migration_incentive = $2,
      WHERE subscription_id = $1 AND status = 'active'
    `, [subscriptionId, JSON.stringify(incentive)]);
  }

  async acceptMigration(subscriptionId: string): Promise<void> {
    await this.pool.query(`
      UPDATE grandfathered_subscriptions
      SET status = 'migrated', migration_response = 'accepted', migration_response_date = NO
      WHERE subscription_id = $1 AND status = 'active'
    `, [subscriptionId]);
  }
}
```

SECTION-46-DUAL-ADMIN-APPROVAL

SECTION 46: DUAL-ADMIN MIGRATION APPROVAL (v4.16.0)

Version: 4.16.0 | Two-person approval for production database migrations

---

46.1 DUAL-ADMIN APPROVAL PRINCIPLES

- **Production migrations require 2 approvals** (configurable)
- **Requestor cannot self-approve**
- **Complete audit trail** of all decisions
- **Configurable policies** per tenant/environment

46.2 DATABASE SCHEMA

packages/infrastructure/migrations/046_dual_admin_approval.sql

```sql
-- ============================================================================
-- RADIANT v4.16.0 - Dual-Admin Migration Approval
-- ============================================================================

-- Migration Approval Requests
CREATE TABLE migration_approval_requests (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),

    migration_name VARCHAR(255) NOT NULL,
    migration_version VARCHAR(50) NOT NULL,
    migration_checksum VARCHAR(64) NOT NULL,
    migration_sql TEXT NOT NULL,

    environment VARCHAR(20) NOT NULL CHECK (environment IN ('development', 'staging', 'produ

    requested_by UUID NOT NULL REFERENCES administrators(id),
    requested_at TIMESTAMPTZ DEFAULT NOW(),
    request_reason TEXT,

    status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (status IN (
        'pending', 'approved', 'rejected', 'executed', 'failed', 'cancelled'
    )),

    approvals_required INTEGER NOT NULL DEFAULT 2,
    approvals_received INTEGER NOT NULL DEFAULT 0,

    executed_at TIMESTAMPTZ,
    executed_by UUID REFERENCES administrators(id),
    execution_time_ms INTEGER,
    execution_error TEXT,
    rollback_sql TEXT,

    metadata JSONB DEFAULT '{}',
```

```
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_migration_approval_tenant ON migration_approval_requests(tenant_id);
CREATE INDEX idx_migration_approval_status ON migration_approval_requests(status);
CREATE INDEX idx_migration_approval_env ON migration_approval_requests(environment);

-- Individual Approvals
CREATE TABLE migration_approvals (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    request_id UUID NOT NULL REFERENCES migration_approval_requests(id) ON DELETE CASCADE,

    admin_id UUID NOT NULL REFERENCES administrators(id),
    decision VARCHAR(20) NOT NULL CHECK (decision IN ('approved', 'rejected')),
    reason TEXT,

    reviewed_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(request_id, admin_id)
);

CREATE INDEX idx_migration_approvals_request ON migration_approvals(request_id);

-- Approval Policies
CREATE TABLE migration_approval_policies (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    environment VARCHAR(20) NOT NULL,

    approvals_required INTEGER NOT NULL DEFAULT 2,
    self_approval_allowed BOOLEAN DEFAULT FALSE,
    auto_approve_development BOOLEAN DEFAULT TRUE,

    allowed_approvers UUID[] DEFAULT '{}',
    required_approvers UUID[],

    notification_channels JSONB DEFAULT '{}',
    escalation_after_hours INTEGER DEFAULT 24,

    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tenant_id, environment)
);
```

```sql
-- Trigger: Update approval count
CREATE OR REPLACE FUNCTION update_approval_count()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.decision = 'approved' THEN
        UPDATE migration_approval_requests
        SET approvals_received = approvals_received + 1,
            status = CASE WHEN approvals_received + 1 >= approvals_required THEN 'approved'
            updated_at = NOW()
        WHERE id = NEW.request_id;
    ELSIF NEW.decision = 'rejected' THEN
        UPDATE migration_approval_requests SET status = 'rejected', updated_at = NOW() WHERE
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER migration_approval_update
    AFTER INSERT ON migration_approvals
    FOR EACH ROW EXECUTE FUNCTION update_approval_count();

-- Enable RLS
ALTER TABLE migration_approval_requests ENABLE ROW LEVEL SECURITY;
ALTER TABLE migration_approvals ENABLE ROW LEVEL SECURITY;
ALTER TABLE migration_approval_policies ENABLE ROW LEVEL SECURITY;

CREATE POLICY mar_tenant ON migration_approval_requests
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY map_tenant ON migration_approval_policies
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
```

46.3 MIGRATION APPROVAL SERVICE

packages/functions/src/services/migration-approval.ts

```typescript
/**
 * Dual-Admin Migration Approval Service
 * @version 4.16.0
 */

import { Pool } from 'pg';
import crypto from 'crypto';

export class MigrationApprovalService {
```

```
constructor(private pool: Pool) {}

async createRequest(
  tenantId: string,
  adminId: string,
  migrationName: string,
  migrationVersion: string,
  migrationSql: string,
  environment: string,
  reason?: string
): Promise<{ id: string; status: string; approvalsRequired: number }> {
  // Get policy
  const policyResult = await this.pool.query(`
    SELECT * FROM migration_approval_policies
    WHERE tenant_id = $1 AND environment = $2 AND is_active = TRUE
  `, [tenantId, environment]);

  let approvalsRequired = environment === 'production' ? 2 : (environment === 'staging' ?
  if (policyResult.rows.length > 0) {
    const policy = policyResult.rows[0];
    approvalsRequired = policy.approvals_required;
    if (environment === 'development' && policy.auto_approve_development) approvalsRequire
  }

  const checksum = crypto.createHash('sha256').update(migrationSql).digest('hex');

  const result = await this.pool.query(`
    INSERT INTO migration_approval_requests (
      tenant_id, migration_name, migration_version, migration_checksum,
      migration_sql, environment, requested_by, request_reason,
      approvals_required, status
    ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10)
    RETURNING id, status, approvals_required
  `, [tenantId, migrationName, migrationVersion, checksum, migrationSql, environment, admi

  return result.rows[0];
}

async submitApproval(
  requestId: string,
  adminId: string,
  decision: 'approved' | 'rejected',
  reason?: string
): Promise<{ success: boolean; requestStatus: string; canExecute: boolean }> {
  const client = await this.pool.connect();
```

```
try {
  await client.query('BEGIN');

  // Get request
  const requestResult = await client.query(`SELECT * FROM migration_approval_requests WH
  if (requestResult.rows.length === 0) throw new Error('Request not found');
  const request = requestResult.rows[0];

  if (request.status !== 'pending') throw new Error(`Request is already ${request.status

  // Check self-approval
  const policyResult = await client.query(`
    SELECT self_approval_allowed FROM migration_approval_policies
    WHERE tenant_id = $1 AND environment = $2 AND is_active = TRUE
  `, [request.tenant_id, request.environment]);

  const selfApprovalAllowed = policyResult.rows[0]?.self_approval_allowed ?? false;
  if (request.requested_by === adminId && !selfApprovalAllowed) {
    throw new Error('Self-approval is not allowed for this environment');
  }

  // Check duplicate
  const existingApproval = await client.query(`
    SELECT id FROM migration_approvals WHERE request_id = $1 AND admin_id = $2
  `, [requestId, adminId]);
  if (existingApproval.rows.length > 0) throw new Error('You have already submitted an a

  // Insert approval
  await client.query(`
    INSERT INTO migration_approvals (request_id, admin_id, decision, reason)
    VALUES ($1, $2, $3, $4)
  `, [requestId, adminId, decision, reason]);

  // Get updated status
  const updatedResult = await client.query(`
    SELECT status, approvals_received, approvals_required FROM migration_approval_reques
  `, [requestId]);
  const updated = updatedResult.rows[0];

  await client.query('COMMIT');

  return {
    success: true,
    requestStatus: updated.status,
    canExecute: updated.status === 'approved'
  };
```

```
  } catch (error) {
    await client.query('ROLLBACK');
    throw error;
  } finally {
    client.release();
  }
}

async executeMigration(requestId: string, adminId: string): Promise<{ success: boolean; er
  const client = await this.pool.connect();

  try {
    await client.query('BEGIN');

    const requestResult = await client.query(`
      SELECT * FROM migration_approval_requests WHERE id = $1 AND status = 'approved' FOR
    `, [requestId]);
    if (requestResult.rows.length === 0) throw new Error('Request not found or not approve

    const request = requestResult.rows[0];
    const startTime = Date.now();

    try {
      await client.query(request.migration_sql);

      await client.query(`
        UPDATE migration_approval_requests
        SET status = 'executed', executed_at = NOW(), executed_by = $2, execution_time_ms
        WHERE id = $1
      `, [requestId, adminId, Date.now() - startTime]);

      await client.query('COMMIT');
      return { success: true };
    } catch (execError: any) {
      await client.query('ROLLBACK');
      await this.pool.query(`
        UPDATE migration_approval_requests SET status = 'failed', execution_error = $2, up
      `, [requestId, execError.message]);
      return { success: false, error: execError.message };
    }
  } catch (error) {
    await client.query('ROLLBACK');
    throw error;
  } finally {
    client.release();
  }
```

```
  }

  async getPendingRequests(tenantId: string, adminId: string): Promise<any[]> {
    const result = await this.pool.query(`
      SELECT mar.*, NOT EXISTS (SELECT 1 FROM migration_approvals ma WHERE ma.request_id = m
      FROM migration_approval_requests mar
      WHERE mar.tenant_id = $1 AND mar.status = 'pending' AND mar.requested_by != $2
      ORDER BY mar.created_at DESC
    `, [tenantId, adminId]);
    return result.rows;
  }
}
```

IMPLEMENTATION VERIFICATION CHECKLIST (v4.13.0 - v4.16.0)

Section 43: Billing & Credits (v4.13.0)

- ☐ Migration 043_billing_system.sql applied
- ☐ All 7 subscription tiers seeded
- ☐ credit_pools table created with RLS
- ☐ credit_transactions table created
- ☐ subscriptions table created
- ☐ CreditsService implemented
- ☐ Stripe integration working

Section 44: Storage Billing (v4.14.0)

- ☐ Migration 044_storage_billing.sql applied
- ☐ storage_usage table created
- ☐ storage_pricing configured per tier
- ☐ StorageBillingService implemented
- ☐ S3 usage calculation working
- ☐ Quota warnings sending

Section 45: Versioned Subscriptions (v4.15.0)

- ☐ Migration 045_versioned_subscriptions.sql applied
- ☐ subscription_plan_versions table created

☐ grandfathered_subscriptions table created
☐ get_effective_plan() function working
☐ GrandfatheringService implemented
☐ Migration offers working

Section 46: Dual-Admin Approval (v4.16.0)

☐ Migration 046_dual_admin_approval.sql applied
☐ migration_approval_requests table created
☐ migration_approvals table created
☐ Approval count trigger working
☐ MigrationApprovalService implemented
☐ Self-approval prevention working

RADIANT v4.17.0 - AI-Optimized for Code Generation Version: 4.17.0 | December 2024 | Prompt 32 Total Sections: 47 (0-46)

v4.17.0 AI Code Generation Enhancements:

- Complete RadiantDeployerApp.swift with @main struct
- Package.swift manifest for Swift Package Manager
- Info.plist and entitlements templates
- RADIANT_VERSION constant (replaces hardcoded strings)
- DOMAIN_PLACEHOLDER constant for configuration
- Sendable conformance for all types crossing actor boundaries
- Swift file creation order for AI implementation
- Platform requirements (macOS 13.0+, Swift 5.9+, Xcode 15+)
- AWS CLI path detection (Homebrew ARM64 + Intel + system)
- DeploymentResult.create() factory method
- Enhanced DeploymentError with helpful messages

Previous Fixes (v4.16.1):

- RLS variable standardization (`app.current_tenant_id`)
- Type deduplication (`SelfHostedModelPricing`, `ExternalProviderPricing`)
- Billing tier localization registry entries

END OF RADIANT-PROMPT-32-v4.17.0

PART 5: FEATURE DOCUMENTATION

AI-ETHICS-STANDARDS

AI Ethics Standards Framework

Version: 4.18.3
Last Updated: 2024-12-28

Overview

RADIANT's ethical AI behavior is guided by principles that map to established industry standards. This document provides full transparency into the standards framework and how each ethical principle aligns with recognized AI ethics guidelines.

Industry Standards

Government & Regulatory

NIST AI Risk Management Framework (AI RMF 1.0)

Field	Value
Code	`NIST_AI_RMF`
Full Name	NIST AI Risk Management Framework
Version	1.0
Organization	National Institute of Standards and Technology
Published	January 26, 2023
Status	Required
URL	https://www.nist.gov/itl/ai-risk-management-framework

Description: Comprehensive framework for managing risks in AI systems throughout their lifecycle. Provides guidance on governance, mapping, measurement, and management of AI risks.

Key Functions: - **GOVERN**: Establish AI governance - **MAP**: Map and characterize AI context - **MEASURE**: Assess and analyze AI risks - **MANAGE**: Prioritize and act on AI risks

---

ISO/IEC 42001:2023 AI Management System

Field	Value
Code	`ISO_42001`
Full Name	ISO/IEC 42001:2023 AI Management System
Version	2023
Organization	International Organization for Standardization
Published	December 18, 2023
Status	Required
URL	https://www.iso.org/standard/81230.html

Description: First international AI management system standard. Provides requirements for establishing, implementing, maintaining and continually improving an AI management system.

Key Clauses: - **Clause 4**: Context of the organization - **Clause 5**: Leadership - **Clause 6**: Planning - **Clause 7**: Support - **Clause 8**: Operation - **Clause 9**: Performance evaluation - **Clause 10**: Improvement

---

EU AI Act

Field	Value
Code	`EU_AI_ACT`
Full Name	European Union Artificial Intelligence Act
Version	2024
Organization	European Union
Published	March 13, 2024
Status	Required
URL	https://artificialintelligenceact.eu/

Description: Comprehensive regulatory framework for AI systems in the EU. Establishes risk-based approach with requirements for high-risk AI systems.

Key Articles: - **Article 7**: Special attention to vulnerable groups - **Article 9**: Risk management system - **Article 10**: Data governance - **Article 13**: Transparency obligations - **Article 14**: Human oversight

---

Industry & Academic

IEEE 7000-2021

Field	Value
Code	`IEEE_7000`
Full Name	IEEE 7000-2021 Model Process for Addressing Ethical Concerns
Version	2021
Organization	Institute of Electrical and Electronics Engineers
Published	September 15, 2021
URL	https://standards.ieee.org/ieee/7000/6781/

Description: Standard for addressing ethical concerns during system design. Provides model process for identifying and addressing ethical values.

OECD AI Principles

Field	Value
Code	`OECD_AI`
Full Name	OECD Principles on Artificial Intelligence
Version	2019
Organization	Organisation for Economic Co-operation and Development
Published	May 22, 2019
URL	https://oecd.ai/en/ai-principles

Description: First intergovernmental standard on AI. Promotes AI that is innovative, trustworthy, and respects human rights and democratic values.

UNESCO AI Ethics

Field	Value
Code	`UNESCO_AI`
Full Name	UNESCO Recommendation on the Ethics of AI
Version	2021
Organization	United Nations Educational, Scientific and Cultural Organization
Published	November 23, 2021

Field	Value
URL	https://www.unesco.org/en/artificial-intelligence/recommendation-ethics

Description: First global standard-setting instrument on ethics of AI. Adopted by all 193 Member States.

Additional ISO Standards

ISO/IEC 23894:2023

Field	Value
Code	`ISO_23894`
Full Name	ISO/IEC 23894:2023 AI Risk Management
Version	2023
Organization	International Organization for Standardization
Published	February 1, 2023
URL	https://www.iso.org/standard/77304.html

Description: Guidance on managing risk for organizations using or developing AI systems. Complements ISO 31000 risk management.

ISO/IEC 38507:2022

Field	Value
Code	`ISO_38507`
Full Name	ISO/IEC 38507:2022 Governance of IT - AI
Version	2022
Organization	International Organization for Standardization
Published	April 1, 2022
URL	https://www.iso.org/standard/56641.html

Description: Guidance for governing bodies on AI governance. Extends IT governance to address AI-specific considerations.

Principle-to-Standard Mapping

Each ethical principle in RADIANT maps to specific sections of industry standards:

Love Others

Standard	Section	Requirement
NIST AI RMF	GOVERN 1.2	Organizations should ensure AI systems respect human dignity
ISO/IEC 42001	Clause 5.2	AI policy shall include commitment to human-centered values
Christian Ethics	Matthew 22:39	Love your neighbor as yourself

Golden Rule

Standard	Section	Requirement
NIST AI RMF	MAP 1.1	Intended purpose and context of use are documented
EU AI Act	Article 9	Risk management system shall consider effects on persons
Christian Ethics	Matthew 7:12	Do to others what you would have them do to you

Speak Truth

Standard	Section	Requirement
NIST AI RMF	GOVERN 4.1	Organizational transparency about AI system capabilities and limitations
ISO/IEC 42001	Clause 7.4	Communication shall be truthful and clear
EU AI Act	Article 13	Transparency obligations for high-risk AI systems
Christian Ethics	John 8:32	The truth will set you free

Show Mercy

Standard	Section	Requirement
NIST AI RMF	MEASURE 2.6	AI systems should minimize potential harms
EU AI Act	Article 14	Human oversight to minimize risks
Christian Ethics	Matthew 5:7	Blessed are the merciful

Serve Humbly

Standard	Section	Requirement
NIST AI RMF	GOVERN 1.1	Policies reflect commitment to accountability
ISO/IEC 42001	Clause 5.1	Top management shall demonstrate leadership and commitment
Christian Ethics	Mark 10:45	The greatest among you shall be your servant

Avoid Judgment

Standard	Section	Requirement
NIST AI RMF	MAP 2.3	Scientific integrity and objectivity in AI assessments
EU AI Act	Article 10	Data governance to avoid bias
Christian Ethics	Matthew 7:1	Do not judge, or you too will be judged

Care for Vulnerable

Standard	Section	Requirement
NIST AI RMF	MAP 1.5	Potential impacts on individuals and communities identified
EU AI Act	Article 7	Special attention to vulnerable groups
ISO/IEC 42001	Clause 8.4	Consider impacts on interested parties
Christian Ethics	Matthew 25:40	Whatever you did for the least of these, you did for me

Alignment Levels

Principles map to standards with different alignment levels:

Level	Description	Icon
derived	Principle was directly derived from this standard	
aligned	Principle aligns with this standard's requirements	
supports	Principle supports this standard's goals	
extends	Principle extends beyond this standard	

Database Schema

ai_ethics_standards

Column	Type	Description
`id`	UUID	Primary key
`code`	VARCHAR(50)	Unique standard code (e.g., 'NIST_AI_RMF')
`name`	VARCHAR(255)	Short name
`full_name`	VARCHAR(500)	Complete standard title
`version`	VARCHAR(50)	Version number
`organization`	VARCHAR(255)	Issuing organization
`organization_type`	VARCHAR(50)	government, iso, industry, academic, religious
`description`	TEXT	Summary description
`url`	VARCHAR(500)	Link to official standard
`publication_date`	DATE	When published
`is_mandatory`	BOOLEAN	Required for compliance
`display_order`	INTEGER	UI ordering
`icon`	VARCHAR(50)	Lucide icon name

ai_ethics_principle_standards

Column	Type	Description
`id`	UUID	Primary key
`principle_id`	UUID	FK to ethical_principles
`standard_id`	UUID	FK to ai_ethics_standards

Column	Type	Description
`standard_section`	VARCHAR(100)	Specific section (e.g., 'MAP 1.1')
`standard_requirement`	TEXT	Requirement text
`alignment_level`	VARCHAR(20)	derived, aligned, supports, extends

API Endpoints

GET /admin/ethics/standards

Returns all active AI ethics standards.

Response:

```
{
  "standards": [
    {
      "code": "NIST_AI_RMF",
      "name": "NIST AI RMF",
      "fullName": "NIST AI Risk Management Framework",
      "version": "1.0",
      "organization": "National Institute of Standards and Technology",
      "organizationType": "government",
      "description": "Comprehensive framework for managing risks...",
      "url": "https://www.nist.gov/itl/ai-risk-management-framework",
      "publicationDate": "2023-01-26",
      "isMandatory": true,
      "icon": "Shield"
    }
  ]
}
```

GET /admin/ethics/principles

Returns ethical principles with their standard mappings.

Response:

```
{
  "principles": [
    {
      "principleId": "uuid",
      "name": "Love Others",
      "teaching": "Love your neighbor as yourself",
      "source": "Matthew 22:39",
      "category": "love",
```

```
      "weight": 1.0,
      "standards": [
        {
          "code": "NIST_AI_RMF",
          "name": "NIST AI RMF",
          "fullName": "NIST AI Risk Management Framework",
          "section": "GOVERN 1.2",
          "requirement": "Organizations should ensure AI systems respect human dignity",
          "isMandatory": true
        }
      ]
    }
  ]
}
```

Admin Dashboard

Location: Admin Dashboard → Ethics → Standards

Features

1. **Standards List**: All industry frameworks with metadata
2. **Principle Mapping**: See which standards each principle aligns with
3. **External Links**: Direct links to official standard documents
4. **Mandatory Indicators**: Red badges for required standards
5. **Organization Types**: Color-coded by type (government, ISO, industry, academic, religious)

Related Documentation

- RADIANT Admin Guide - Ethics Section
- Ethical Guardrails Service

PROVIDER-REJECTION-HANDLING

Provider Rejection Handling & Intelligent Fallback

Version: 4.18.3
Last Updated: 2024-12-28

Overview

When an AI provider or self-hosted model rejects a prompt based on their ethics policies (that don't conflict with RADIANT's ethics), the system automatically attempts fallback to alternative models. If all capable models reject the request, the user receives a clear explanation.

How It Works

Rejection Flow

```
1. User submits prompt
2. AGI Brain selects optimal model
3. Model rejects prompt (provider ethics, content policy, etc.)
4. System checks: Does this violate OUR ethics?
     YES → Reject to user with explanation
     NO → Try fallback models
5. Fallback loop (max 3 attempts):
     Select model with lowest rejection rate
     Attempt request
     If success → Return response
6. If all fallbacks fail:
     Reject to user with detailed explanation
```

Key Principles

1. **RADIANT ethics take precedence** - If our ethics block it, no fallback is attempted
2. **Provider ethics don't block us** - Different providers have different policies; we route around them
3. **Users always know** - Every rejection is explained to the user
4. **Learning from patterns** - The system learns which models reject which types of content

---

Rejection Types

Type	Description	Fallback?
`content_policy`	Provider's content policy violation	Yes
`safety_filter`	Safety/moderation filter triggered	Yes
`provider_ethics`	Provider's ethical guidelines differ	Yes

Type	Description	Fallback?
`capability_mismatch`	Model can't handle this request type	Yes
`context_length`	Prompt too long for model	Yes
`moderation`	Pre-flight moderation blocked	Yes
`rate_limit`	Rate limiting (retry later)	Retry
`unknown`	Unknown error	Yes

Fallback Model Selection

Models are selected for fallback based on:

1. **Rejection rate** - Models with lowest historical rejection rates preferred
2. **Required capabilities** - Must have same capabilities as original
3. **Exclusion list** - Previously tried models excluded
4. **Provider diversity** - Prefer different providers for better success chance

Selection Query

```
SELECT model_id, provider_id, rejection_rate
FROM unified_model_registry m
LEFT JOIN model_rejection_stats s ON m.model_id = s.model_id
WHERE m.enabled = true
  AND m.model_id != ALL(excluded_models)
  AND m.capabilities && required_capabilities
ORDER BY COALESCE(s.rejection_rate, 0) ASC
LIMIT 10
```

User Notifications

Notification Types

Rejected Request:

```
Title: Request Could Not Be Completed
Message: The ethical guidelines of available AI providers prevented
         this response. We attempted 3 different AI models.
Suggested Actions:
  - Try rephrasing your request
  - Remove potentially sensitive content
  - Contact administrator
```

Resolved with Fallback:

```
Title: Resolved with Alternative Model
Message: Your request was processed by an alternative AI model
         after the original was unavailable.
```

Think Tank UI

- **Bell icon** with unread count in toolbar
- **Sheet panel** slides out showing all notifications
- **Rejection banners** appear in conversation when relevant
- **Suggested actions** are clickable to help users resolve issues

---

Database Schema

provider_rejections

Column	Type	Description
id	UUID	Primary key
tenant_id	UUID	Multi-tenant isolation
user_id	UUID	User who made request
plan_id	UUID	AGI Brain plan if applicable
model_id	VARCHAR	Model that rejected
provider_id	VARCHAR	Provider ID
rejection_type	VARCHAR	Type of rejection
rejection_message	TEXT	Raw error from provider
radiant_ethics_passed	BOOLEAN	Did it pass our ethics?
fallback_attempted	BOOLEAN	Was fallback tried?
fallback_model_id	VARCHAR	Model that succeeded
fallback_succeeded	BOOLEAN	Did fallback work?
fallback_chain	JSONB	Array of all attempts
final_status	VARCHAR	pending, fallback_success, rejected
final_response_to_user	TEXT	Message shown to user

rejection_patterns

Learns patterns for smarter fallback:

Column	Type	Description
pattern_hash	VARCHAR	Hash of rejection characteristics
trigger_keywords	TEXT[]	Keywords that trigger rejections
trigger_model_ids	TEXT[]	Models that reject this pattern
recommended_fallback_models	TEXT[]	Models that work
success_rate	NUMERIC	Fallback success rate

model_rejection_stats

Per-model rejection statistics:

Column	Type	Description
`model_id`	VARCHAR	Model identifier
`total_requests`	INTEGER	Total requests to model
`total_rejections`	INTEGER	Total rejections
`rejection_rate`	NUMERIC	Computed rejection rate
`content_policy_count`	INTEGER	By type breakdown
`fallback_successes`	INTEGER	Successful fallbacks

API Endpoints

Record Rejection

```
POST /api/internal/rejections
{
  "modelId": "gpt-4",
  "providerId": "openai",
  "rejectionType": "content_policy",
  "rejectionMessage": "Content policy violation",
  "planId": "uuid"
}
```

Get User Notifications

```
GET /api/thinktank/rejections
Response: {
  "notifications": [...],
  "unreadCount": 3
}
```

Mark Notification Read

```
PATCH /api/thinktank/rejections/:id/read
```

Dismiss Notification

```
DELETE /api/thinktank/rejections/:id
```

Service Integration

ProviderRejectionService

```
// Handle rejection with automatic fallback
const result = await providerRejectionService.handleRejectionWithFallback(
  tenantId,
  userId,
  originalModelId,
  providerId,
  rejectionType,
  rejectionMessage,
  async (modelId, providerId) => {
    // Execute request with fallback model
    return await executeRequest(modelId, providerId, prompt);
  },
  planId,
  requiredCapabilities
);

if (result.success) {
  // Request succeeded (possibly with fallback)
  console.log('Handled by:', result.handlingModelId);
  console.log('Used fallback:', result.usedFallback);
} else {
  // All models rejected
  console.log('Rejection reason:', result.rejectionReason);
  console.log('User message:', result.userFacingMessage);
}
```

AGI Brain Integration

The AGI Brain Planner automatically uses rejection handling:

1. Selects optimal model for task
2. If model rejects, calls `providerRejectionService.handleRejectionWithFallback()`
3. If fallback succeeds, continues with new model
4. If all fail, returns rejection response to user

Configuration

Constants

```
const MIN_MODELS_FOR_TASK = 2;   // Minimum models needed
const MAX_FALLBACK_ATTEMPTS = 3; // Maximum fallback tries
```

Model Rejection Thresholds

Models with rejection rates above 30% are deprioritized for initial selection but may still be used as fallbacks.

Admin Dashboard

Rejection Analytics

Location: Admin Dashboard → Analytics → Rejections

Full analytics dashboard for monitoring rejections and informing policy updates.

Summary Cards

- **Total Rejections (30d)** - All rejections in period
- **Fallback Success Rate** - Percentage resolved via fallback
- **Rejected to User** - Requests that failed all fallbacks
- **Flagged Keywords** - Keywords marked for policy review

Tabs

Tab	Purpose
By Provider	See which providers reject most, rejection types, fallback rates
Violation Keywords	Keywords triggering rejections, per-provider breakdown
Flagged Prompts	Full prompt content for policy investigation
Policy Review	Recommendations for pre-filters based on patterns

Viewing Full Prompt Content

Administrators can view the complete rejected prompt to understand why it was rejected:

1. Go to Analytics → Rejections → Flagged Prompts
2. Click "View Full Prompt" on any entry
3. Review detected keywords and rejection reason
4. Decide: Add Pre-Filter, Add Warning, or Dismiss

Adding Pre-Filters

Based on rejection patterns, add pre-filters to RADIANT's ethics:

1. Identify high-frequency rejection keywords

2. Flag keywords for review
3. Investigate sample prompts
4. Add pre-filter rule to block before sending to AI

Database Views

View	Purpose
`rejection_summary_by_provider`	Aggregated stats per provider
`rejection_summary_by_model`	Aggregated stats per model
`top_rejection_keywords`	Most frequent violation keywords

Related Documentation

- AGI Brain Plan System
- AI Ethics Standards
- Model Router Service

USER-RULES-SYSTEM

User Rules System (Memory Rules)

Version: 4.18.3
Last Updated: 2024-12-28

Overview

The User Rules System allows Think Tank users to set persistent personal preferences that govern how the AI responds to them. Similar to Windsurf policies but for end users, these rules are automatically applied to every AI interaction.

Key Concepts

Rule Types

Type	Description	Example
restriction	Things the AI must NOT do	"Do not discuss religion"
preference	Things the AI SHOULD do	"Acknowledge uncertainty"
format	How responses should be structured	"Use bullet points"

Type	Description	Example
source	Citation requirements	"Always cite sources"
tone	Communication style	"Be concise"
topic	Topic-specific rules	"Add health disclaimers"
privacy	Personal data handling	"Protect my privacy"
accessibility	Readability preferences	"Use simple language"

Rule Sources

- **user_created**: User typed the rule manually
- **preset_added**: Added from the preset library
- **ai_suggested**: AI suggested based on feedback patterns
- **imported**: Imported from another source

Think Tank UI

Location: Think Tank → My Rules
URL: `/thinktank/my-rules`

My Rules Tab

View and manage your personal rules:

- **Toggle**: Enable/disable individual rules
- **Edit**: Modify rule text
- **Delete**: Remove rules
- **Stats**: See how many times each rule was applied

Add from Presets Tab

Browse and add pre-seeded rules:

Popular Rules - Most commonly used rules by Think Tank users

Categories: - Privacy & Safety - Sources & Citations - Response Format - Tone & Style - Accessibility - Topic Preferences - Advanced

---

Pre-seeded Preset Rules

Privacy & Safety

Rule	Description
Protect my privacy	Prevents personal references or assumptions
No religious content	Filters out religious discussions
No political content	Keeps responses politically neutral

Sources & Citations

Rule	Description
Always cite sources	Includes verifiable sources for facts
Prefer academic sources	Prioritizes peer-reviewed content
Include source dates	Adds publication dates for recency

Response Format

Rule	Description
Be concise	Produces shorter, focused responses
Use lists for clarity	Organizes with bullets/numbers
Use headings	Adds section headers to long content
Comment code	Documents code examples

Tone & Style

Rule	Description
Professional tone	Business-appropriate style
Casual tone	Relaxed, conversational style
Simple explanations	Accessible without oversimplifying

Advanced

Rule	Description
Acknowledge uncertainty	States limitations honestly
Clarify before answering	Confirms question understanding
Show multiple viewpoints	Balanced coverage of debates

How Rules Are Applied

Application Flow

1. User sends message to Think Tank
2. AGI Brain generates plan with pre-prompt selection
3. `prepromptLearningService.selectPreprompt()` is called
4. Service fetches user rules via `userRulesService.getRulesForPrompt()`
5. Rules are formatted and appended to the system prompt
6. Rule application is logged for tracking

Prompt Injection Format

Rules are injected into the system prompt in categorized sections:

```
## User Preferences
The user has set the following rules for how you should respond:

**Restrictions (Must Follow):**
- Do not discuss religious topics...

**Source Requirements:**
- Always provide sources and citations...

**Format Preferences:**
- Keep responses concise...
```

Priority

- Restrictions are always enforced first (highest priority)
- Higher priority numbers (0-100) take precedence
- Conflicting rules resolved by priority

Database Schema

user_memory_rules

Column	Type	Description
`id`	UUID	Primary key
`tenant_id`	UUID	Multi-tenant isolation
`user_id`	UUID	Rule owner
`rule_text`	TEXT	Full rule content
`rule_summary`	VARCHAR	Short display text
`rule_type`	VARCHAR	restriction, preference, etc.
`priority`	INTEGER	0-100, higher = more important
`source`	VARCHAR	user_created, preset_added, etc.

Column	Type	Description
is_active	BOOLEAN	Enable/disable
apply_to_preprompts	BOOLEAN	Apply to system prompts
apply_to_synthesis	BOOLEAN	Apply during synthesis
times_applied	INTEGER	Usage counter

preset_user_rules

Column	Type	Description
id	UUID	Primary key
rule_text	TEXT	Full rule content
rule_summary	VARCHAR	Short display text
description	TEXT	User-facing explanation
rule_type	VARCHAR	Rule category
category	VARCHAR	UI grouping
icon	VARCHAR	Lucide icon name
is_popular	BOOLEAN	Show in popular section
min_tier	INTEGER	Subscription tier requirement

API Endpoints

User Rules

```
GET    /api/thinktank/user-rules           - Get user's rules
POST   /api/thinktank/user-rules           - Create new rule
PATCH  /api/thinktank/user-rules/:id       - Update rule
DELETE /api/thinktank/user-rules/:id       - Delete rule
PATCH  /api/thinktank/user-rules/:id/toggle - Enable/disable rule
```

Presets

```
GET  /api/thinktank/user-rules/presets     - Get preset categories
POST /api/thinktank/user-rules/add-preset  - Add preset to user rules
```

Internal (Service Layer)

```
getRulesForPrompt(tenantId, userId, domainId?, mode?)
  → Returns formatted rules for prompt injection
```

Service Integration

user-rules.service.ts

```
// Get rules formatted for prompt injection
const rules = await userRulesService.getRulesForPrompt(
  tenantId,
  userId,
  domainId,     // Optional: filter by domain
  mode          // Optional: filter by orchestration mode
);

// Returns:
{
  rules: UserMemoryRule[],
  formattedForPrompt: string,
  ruleCount: number,
  hasRestrictions: boolean,
  hasSourceRequirements: boolean
}
```

preprompt-learning.service.ts Integration

The preprompt service automatically fetches and applies user rules:

```
// In selectPreprompt()
const userRules = await userRulesService.getRulesForPrompt(
  request.tenantId,
  request.userId,
  request.detectedDomainId,
  request.orchestrationMode
);

// Append to rendered preprompt
rendered.full = rendered.full + userRules.formattedForPrompt;
```

Best Practices

Writing Effective Rules

1. **Be Specific**: "Always cite sources with URLs" vs "cite sources"
2. **Use Positive Framing**: "Use bullet points" vs "Don't write paragraphs"
3. **One Rule Per Preference**: Easier to toggle and track

Rule Limits

- Maximum 50 rules per user

- Maximum 1000 characters per rule text
- Inactive rules don't count toward limits

When to Use Presets vs Custom

- **Presets**: Common preferences with proven effectiveness
- **Custom**: Unique personal requirements

Memory Categories

Each memory/rule is categorized by **what it IS**, enabling better organization and future expansion.

Category Hierarchy

Top-Level	Sub-Categories	Description
Instruction	format, tone, source	Direct instructions for AI behavior
Preference	style, detail	Preferences that guide (not mandate) behavior
Context	personal, work, project	Background information about the user
Knowledge	fact, definition, procedure	Facts and information to remember
Constraint	topic, privacy, safety	Hard limits that must be followed
Goal	learning, productivity	User objectives and desired outcomes

Category Codes

```
instruction              # Direct instructions
  instruction.format     # How to structure responses
  instruction.tone       # Communication style
  instruction.source     # Citation requirements

preference               # Preferences
  preference.style       # Writing style preferences
  preference.detail      # Detail level preferences

context                  # User context
```

```
  context.personal       # Personal information
  context.work           # Professional context
  context.project        # Project-specific info

knowledge                # Knowledge to remember
  knowledge.fact         # Specific facts
  knowledge.definition   # Terms and meanings
  knowledge.procedure    # How to do things

constraint               # Hard limits
  constraint.topic       # Topics to avoid
  constraint.privacy     # Privacy rules
  constraint.safety      # Safety limitations

goal                     # User goals
  goal.learning          # Learning objectives
  goal.productivity      # Efficiency goals
```

Database Schema: memory_categories

Column	Type	Description
id	UUID	Primary key
code	VARCHAR	Unique category code (e.g., 'instruction.format')
name	VARCHAR	Display name
parent_id	UUID	Parent category for hierarchy
level	INTEGER	1=top-level, 2=sub-category
path	VARCHAR	Materialized path (e.g., 'instruction.format')
icon	VARCHAR	Lucide icon name
color	VARCHAR	Tailwind color class
is_system	BOOLEAN	System categories cannot be deleted
is_expandable	BOOLEAN	Can users add sub-categories?

API Methods

```javascript
// Get category tree
const tree = await userRulesService.getMemoryCategories();
// Returns: { categories, topLevel, byCode }

// Get memories grouped by category
const grouped = await userRulesService.getMemoriesByCategory(
  tenantId,
  userId,
  categoryCode  // Optional: filter to specific category
);
```

Future Expansion

The category system is designed for expansion: - **Custom Categories**: Users can create sub-categories under expandable parents - **Category Inheritance**: Rules can inherit from parent categories - **Category-Specific Behavior**: Different application logic per category - **Cross-Category Rules**: Rules that span multiple categories

---

Related Documentation

- Pre-Prompt Learning System
- Think Tank Documentation
- AGI Brain Plan System

PREPROMPT-LEARNING-SYSTEM

Pre-Prompt Learning System

Version: 4.18.3
Last Updated: 2024-12-28

Overview

The Pre-Prompt Learning System tracks, evaluates, and learns from the effectiveness of pre-prompts (system prompts) used by the AGI Brain. Instead of blaming pre-prompts for all failures, it uses **attribution analysis** to understand what factor actually caused issues - whether it was the pre-prompt, model selection, orchestration mode, workflow, or domain detection.

Key Concepts

Attribution Analysis

When users provide feedback, the system doesn't just record whether the response was good or bad. It analyzes the full context to determine **what factor was most responsible**:

Factor	Description	When Blamed
Pre-prompt	System instructions were wrong	Tone, format, or approach mismatch
Model	AI model selection was inappropriate	Model lacks capability for task

Factor	Description	When Blamed
Mode	Orchestration mode was wrong	Extended thinking when simple needed
Workflow	Workflow pattern didn't fit	Multi-step when single response needed
Domain	Domain detection was incorrect	Medical advice for cooking question
Other	External factors	User unclear, ambiguous request

Learning Weights

Each pre-prompt template has configurable weights that affect selection:

```
Final Score = Base + (Domain × DomainWeight) + (Mode × ModeWeight) +
              (Model × ModelWeight) + (Complexity × ComplexityWeight) +
              (TaskType × TaskTypeWeight) + FeedbackAdjustment
```

Weight	Default	Description
`baseEffectivenessScore`	0.5	Starting score
`domainWeight`	0.2	Bonus for matching domain
`modeWeight`	0.2	Bonus for matching mode
`modelWeight`	0.2	Bonus for compatible model
`complexityWeight`	0.15	Bonus for complexity match
`taskTypeWeight`	0.15	Bonus for task type match
`feedbackWeight`	0.1	Historical feedback influence

Exploration vs Exploitation

The system balances **exploitation** (using best-performing templates) with **exploration** (trying other templates to gather learning data):

- **Exploration Rate**: Percentage of requests where a non-optimal template is chosen
- **Default**: 10% exploration, decays over time
- **Minimum**: 1% to ensure continued learning

Admin Dashboard

Location: Admin Dashboard → Orchestration → Pre-Prompts
URL: `/orchestration/preprompts`

Overview Tab

- **Key Metrics**: Templates, Uses, Avg Rating, Thumbs Up Rate, Feedback Count
- **Attribution Pie Chart**: Visual breakdown of what gets blamed
- **Top Performing Templates**: Best-rated templates by feedback
- **Templates Needing Attention**: Low performers requiring adjustment

Templates Tab

- View all pre-prompt templates
- See usage statistics and success rates
- Adjust weights via slider interface
- View applicable modes and domains

Attribution Tab

- Detailed attribution breakdown
- Historical analysis of what factors contribute to success/failure
- Learning sample count

Feedback Tab

- Recent user feedback with ratings
- Attribution labels for each feedback
- Feedback text and timestamps

Pre-Prompt Templates

Default Templates

Template	Modes	Use Case
`standard_reasoning`	thinking, chain_of_thought	General questions
`extended_thinking`	extended_thinking	Complex reasoning
`coding_expert`	coding	Code generation
`creative_writing`	creative	Creative content
`research_synthesis`	research, analysis	Research tasks
`multi_model_consensus`	multi_model, self_consistency	Ensemble queries
`domain_expert`	all	Domain-specific expertise

Template Variables

Templates support `{{variable}}` placeholders:

Variable	Source	Example
{{domain_name}}	Domain detection	“Medicine”
{{domain_confidence}}	Detection confidence	“85”
{{subspecialty_name}}	Subspecialty	“Cardiology”
{{field_name}}	Field	“Healthcare”
{{complexity}}	Prompt analysis	“complex”
{{task_type}}	Task detection	“reasoning”
{{key_topics}}	Extracted topics	“heart, ECG, diagnosis”
{{model_role}}	For multi-model	“primary”
{{proficiencies}}	Domain proficiencies	“reasoning_depth: 9”

Database Schema

preprompt_templates

Stores reusable pre-prompt patterns.

Column	Type	Description
template_code	VARCHAR	Unique identifier
system_prompt	TEXT	Main prompt text
context_template	TEXT	Context with variables
applicable_modes	TEXT[]	Valid orchestration modes
base_effectiveness_score	DECIMAL	Base selection score
*_weight	DECIMAL	Selection weight factors
total_uses	INTEGER	Usage count
avg_feedback_score	DECIMAL	Average rating

preprompt_instances

Tracks actual pre-prompts used in plans.

Column	Type	Description
plan_id	UUID	Link to AGI plan
template_id	UUID	Template used
full_preprompt	TEXT	Rendered pre-prompt
model_id	VARCHAR	Model used
orchestration_mode	VARCHAR	Mode used
detected_domain_id	VARCHAR	Domain detected
response_quality_score	DECIMAL	Verification score

preprompt_feedback

User feedback with attribution.

Column	Type	Description
`instance_id`	UUID	Pre-prompt instance
`rating`	INTEGER	1-5 rating
`thumbs_up`	BOOLEAN	Simple feedback
`issue_attribution`	VARCHAR	What was blamed
`issue_attribution_confidence`	DECIMAL	Attribution confidence
`feedback_text`	TEXT	User comments

preprompt_attribution_scores

Learning data per template/factor combination.

Column	Type	Description
`template_id`	UUID	Template
`factor_type`	VARCHAR	model/mode/domain/etc
`factor_value`	VARCHAR	Specific value
`success_correlation`	DECIMAL	-1 to 1 correlation
`sample_size`	INTEGER	Data points
`confidence`	DECIMAL	Score confidence

API Endpoints

Dashboard

`GET /api/admin/preprompts/dashboard`

Returns dashboard data including metrics, attribution, top/low templates, recent feedback.

Templates

```
GET /api/admin/preprompts/templates
GET /api/admin/preprompts/templates/:id
PATCH /api/admin/preprompts/templates/:id/weights
```

Feedback

```
POST /api/admin/preprompts/feedback
GET /api/admin/preprompts/feedback/recent
```

Learning Config

```
GET /api/admin/preprompts/config
PATCH /api/admin/preprompts/config/:key
```

Integration with AGI Brain

The pre-prompt system integrates with `agi-brain-planner.service.ts`:

1. **Plan Generation**: Calls `prepromptLearningService.selectPreprompt()`
2. **Template Selection**: Scores templates based on context
3. **Variable Rendering**: Fills in `{{variables}}` from plan data
4. **Instance Tracking**: Records which template was used
5. **Feedback Loop**: User feedback updates attribution scores

Code Example

```typescript
const prepromptResult = await prepromptLearningService.selectPreprompt({
  planId,
  tenantId,
  userId,
  orchestrationMode,
  modelId: primary.modelId,
  detectedDomainId: domainResult?.primary_domain?.domain_id,
  taskType: promptAnalysis.taskType,
  complexity: promptAnalysis.complexity,
  variables: {
    domain_name: domainResult?.primary_domain?.domain_name || 'general',
    complexity: promptAnalysis.complexity,
    // ... more variables
  },
});

plan.systemPrompt = prepromptResult.renderedPreprompt.full;
```

Best Practices

When to Adjust Weights

1. **Low feedback scores**: If a template consistently scores below 3.5
2. **High blame rate**: If pre-prompt is blamed >25% of the time
3. **Mode mismatch**: If template works well in some modes but not others

Weight Adjustment Guidelines

Situation	Adjustment
Template works better with specific models	Increase `modelWeight`
Template is mode-sensitive	Increase `modeWeight`
Domain expertise is critical	Increase `domainWeight`
Historical feedback is reliable	Increase `feedbackWeight`

Monitoring Recommendations

- Check attribution distribution weekly
- Review low-performing templates monthly
- Monitor exploration rate effectiveness
- Track thumbs-up rate trends

Related Documentation

- AGI Brain Plan System
- Orchestration Modes
- Domain Taxonomy
- Admin Guide - Orchestration

ORCHESTRATION-METHODS

RADIANT Orchestration Methods Reference

Version: 4.18.0 Last Updated: 2024-12-28

Related Documentation

- Specialty Ranking System - Domain-specific model proficiency rankings
- Domain Taxonomy - Hierarchical domain detection
- AGI Brain Planner - Real-time planning system

Overview

RADIANT's orchestration system provides **17 reusable methods** that can be composed into **49 workflow patterns**. Each method is parameterized and can receive streams from previous methods in the pipeline.

Architecture

```
   Method 1            Method 2            Method 3
  (Generator)           (Critic)          (Synthesizer)

 Input: prompt       Input: stream       Input: stream
 Output: stream      Output: stream      Output: final
```

Stream Chaining

- Each method receives output from previous method(s) via `{{response}}` or `{{responses}}` template variables
- Methods can depend on multiple previous steps (`dependsOnSteps[]`)
- Parallel execution supported with `parallelExecution.enabled`

Output Stream Modes (NEW)

When a method uses N models, you can control how many streams come out:

Mode	Output	Description
`single`	1 stream	Synthesized result → `{{response}}` (default)
`all`	N streams	All model outputs → `{{responses}}` array
`top_n`	1-N streams	Best N by confidence → `{{responses}}` array
`threshold`	0-N streams	Only above confidence threshold → `{{responses}}` array

```
parallelExecution: {
  enabled: true,
  models: ['openai/o1', 'claude-3-5-sonnet', 'deepseek-reasoner'],
  outputMode: 'all',              // Pass all 3 streams to next step
  preserveModelAttribution: true  // Include model ID with each stream
}
```

Example: 3 models → 3 output streams

```
   Model 1            Model 2            Model 3
   (o1)              (Claude)          (DeepSeek)
```

```
{{responses}} = [
  { modelId: 'o1', response: '...', confidence: 0.92 },
  { modelId: 'claude', response: '...', confidence: 0.88 },
  { modelId: 'deepseek', response: '...', confidence: 0.85 }
]
```

```
Next Step
SYNTHESIZE_RESPONSES
receives 3 streams
```

Methods by Category

1. Generation Methods

GENERATE_RESPONSE **Purpose**: Generate a response to a prompt using specified model

Parameter	Type	Default	Description
`temperature`	number	0.7	Creativity/randomness (0-2)
`max_tokens`	integer	4096	Maximum output tokens

Prompt Template:

```
Generate a response to: {{prompt}}

Context: {{context}}
```

Recommended Models: Claude 3.5 Sonnet, GPT-4o, DeepSeek Chat

GENERATE_WITH_COT **Purpose**: Generate response using chain-of-thought reasoning

Parameter	Type	Default	Description
`temperature`	number	0.3	Lower for consistency
`max_tokens`	integer	8192	Extended for reasoning
`thinking_budget`	integer	2000	Tokens for reasoning

Prompt Template:

```
Think through this step-by-step before answering:

{{prompt}}

Show your reasoning, then provide your answer.
```

Recommended Models: OpenAI o1, DeepSeek Reasoner, Claude 3.5 Sonnet

---

REFINE_RESPONSE **Purpose**: Improve a response based on feedback

Parameter	Type	Default	Description
`refinement_focus`	string	"all"	Focus area: all, clarity, accuracy, completeness
`preserve_structure`	boolean	true	Keep original structure

Input Stream: `{{response}}` - Previous response to refine **Input Stream**: `{{feedback}}` - Critique or feedback

Prompt Template:

```
Improve this response based on the feedback:

Original Response: {{response}}

Feedback: {{feedback}}

Provide an improved response that addresses all feedback while maintaining the good parts.
```

---

2. Evaluation Methods

CRITIQUE_RESPONSE **Purpose**: Critically evaluate a response for flaws and improvements

Parameter	Type	Default	Description
`focus_areas`	array	["accuracy", "completeness", "clarity", "logic"]	What to evaluate
`severity_threshold`	string	"medium"	Minimum severity to report: low, medium, high

Input Stream: `{{response}}` - Response to critique

Prompt Template:

```
Critically evaluate this response:

Original Question: {{original_prompt}}
Response: {{response}}

Identify:
1. Factual errors
2. Logical flaws
3. Missing information
4. Clarity issues

For each issue, rate severity (low/medium/high) and suggest fixes.
```

Recommended Models: OpenAI o1, Claude 3.5 Sonnet

JUDGE_RESPONSES **Purpose**: Compare and judge multiple responses to select the best

Parameter	Type	Default	Description
`evaluation_mode`	enum	"pairwise"	pointwise, pairwise, listwise
`criteria`	array	["accuracy", "helpfulness", "clarity", "completeness"]	Evaluation criteria

Input Stream: `{{responses}}` - Array of responses to judge

Prompt Template:

```
Judge these responses to the question:

Question: {{original_prompt}}

{{#each responses}}Response {{@index}}: {{this}}

{{/each}}

Evaluate each on: {{criteria}}

Output: BEST: [number], SCORE: [0-1], REASONING: [explanation]
```

Output: `{ best: number, score: number, reasoning: string }`

VERIFY_FACTS **Purpose**: Extract and verify factual claims in a response

Parameter	Type	Default	Description
`extraction_method`	string	"explicit"	How to find claims: explicit, implicit, all
`verification_depth`	string	"thorough"	Verification level: quick, standard, thorough

Input Stream: `{{response}}` - Response to verify

Prompt Template:

```
Extract all factual claims from this response and verify each:

Response: {{response}}

For each claim:
1. State the claim
2. Verify if true/false/uncertain
3. Provide evidence or reasoning
4. Confidence level
```

GENERATE_CHALLENGE **Purpose**: Challenge a response by arguing the opposite position

Parameter	Type	Default	Description
`challenge_intensity`	string	"moderate"	How aggressive: mild, moderate, aggressive
`focus`	string	"weakest_points"	What to challenge: all, weakest_points, assumptions

Input Stream: `{{response}}` - Response to challenge

DEFEND_POSITION **Purpose**: Defend a response against challenges

Parameter	Type	Default	Description
`defense_strategy`	string	"address_all"	Strategy: address_all, prioritize, concede_gracefully
`concede_valid`	boolean	true	Acknowledge valid challenges

Input Streams: - `{{response}}` - Original response - `{{challenge}}` - Challenge to defend against

---

SELF_REFLECT **Purpose**: AI reflects on its own response to identify improvements

Parameter	Type	Default	Description
`reflection_depth`	string	"thorough"	Depth: quick, standard, thorough
`aspects`	array	["accuracy", "completeness", "clarity"]	What to reflect on

Input Stream: `{{response}}` - Response to reflect on

---

3. Synthesis Methods

SYNTHESIZE_RESPONSES **Purpose**: Combine best parts from multiple responses

Parameter	Type	Default	Description
`combination_strategy`	string	"best_parts"	Strategy: best_parts, merge, weighted
`conflict_resolution`	string	"majority"	How to resolve conflicts: majority, primary, newest

Input Stream: `{{responses}}` - Array of responses with model attribution

Prompt Template:

```
Synthesize these responses into one superior response:

Question: {{original_prompt}}

{{#each responses}}Response from {{model}}: {{content}}

{{/each}}

Create a response that:
1. Takes the best, most accurate parts from each
2. Resolves any conflicts
3. Is comprehensive and well-organized
```

BUILD_CONSENSUS **Purpose**: Identify points of agreement across multiple responses

Parameter	Type	Default	Description
`consensus_threshold`	number	0.7	Minimum agreement ratio (0-1)
`include_disputed`	boolean	true	Include disputed points with caveats

Input Stream: `{{responses}}` - Array of responses

---

4. Routing Methods

DETECT_TASK_TYPE **Purpose**: Analyze prompt to determine task type and complexity

Parameter	Type	Default	Description
`task_categories`	array	[“coding”, “reasoning”, “creative”, “factual”, “math”, “research”]	Categories to detect

Output:

```
{
  "taskType": "coding",
  "complexity": "complex",
  "requiredCapabilities": ["code_generation", "debugging"],
  "recommendedApproach": "chain_of_thought"
}
```

Recommended Models: GPT-4o-mini, Claude 3.5 Haiku (fast, cheap)

SELECT_BEST_MODEL **Purpose**: Choose the optimal model for a given task

Parameter	Type	Default	Description
`consider_cost`	boolean	true	Factor in cost
`consider_latency`	boolean	true	Factor in speed
`quality_priority`	number	0.7	Quality vs cost tradeoff (0-1)

Implementation: `code` - `model-selection-service.selectBestModel`

---

5. Reasoning Methods

DECOMPOSE_PROBLEM **Purpose**: Break down a complex problem into sub-problems

Parameter	Type	Default	Description
`max_subproblems`	integer	5	Maximum sub-problems
`decomposition_strategy`	string	"functional"	Strategy: functional, temporal, hierarchical

Prompt Template:

```
Decompose this problem into smaller sub-problems:

Problem: {{prompt}}

1. Identify independent components
2. Order by dependency
3. Estimate complexity of each
4. Return structured decomposition
```

Recommended Models: OpenAI o1, Claude 3.5 Sonnet

---

6. Aggregation Methods

MAJORITY_VOTE **Purpose**: Select the most common answer from multiple responses

Parameter	Type	Default	Description
`vote_method`	string	"exact_match"	Matching: exact_match, semantic, fuzzy
`tie_breaker`	string	"first"	Tie resolution: first, random, highest_confidence

Implementation: `code` - `aggregation-service.majorityVote`

WEIGHTED_AGGREGATE **Purpose**: Combine responses weighted by confidence/expertise

Parameter	Type	Default	Description
`weight_by`	string	"confidence"	Weight source: confidence, expertise, recency
`normalize`	boolean	true	Normalize weights to sum to 1

Implementation: `code` - `aggregation-service.weightedAggregate`

Workflow Patterns (49 Total)

Categories

Category	Count	Description
Adversarial & Validation	2	Security testing, vulnerability discovery
Debate & Deliberation	3	Multi-perspective analysis

Category	Count	Description
Judge & Critic	3	Quality evaluation and improvement
Ensemble & Aggregation	3	Multi-model synthesis
Reflection & Self-Improvement	3	Iterative refinement
Verification & Fact-Checking	2	Accuracy validation
Multi-Agent Collaboration	2	Team-based problem solving
Reasoning Enhancement	9	CoT, ToT, ReAct, etc.
Model Routing Strategies	4	Optimal model selection
Domain-Specific Orchestration	4	Domain expertise routing
Cognitive Frameworks	14	First Principles, Systems Thinking, etc.

Pattern Quick Reference

Code	Name	Models	Latency	Quality Improvement
`CoT`	Chain-of-Thought	1	Medium	+20-40% on math/logic
`SCMR`	Majority Vote	3+	Medium	+15-25% accuracy
`ISFR`	Self-Refine Loop	1	High	+20-30% per iteration
`MDA`	Multi-Agent Debate	3+	Very High	+30-45% consensus
`CASCADE`	Cascade	2+	Variable	40-60% cost reduction
`ToT`	Tree-of-Thoughts	1	Very High	4%→74% on puzzles

Metrics Captured

For each method execution:

Metric	Description
`latencyMs`	Execution time in milliseconds
`costCents`	Cost in cents
`tokensUsed`	Input + output tokens
`modelUsed`	Model ID used
`qualityScore`	Auto-assessed quality (0-1)
`wasParallel`	Whether parallel execution was used

Metric	Description
`parallelResults`	Individual model results if parallel
`iteration`	Iteration number for iterative methods

Aggregated Metrics (per workflow)

Metric	Description
`avgQualityScore`	Rolling average quality
`avgLatencyMs`	Average latency
`avgCostCents`	Average cost
`executionCount`	Total executions
`successRate`	Completion rate

Admin Configuration

Per-Tenant Customization

Admins can customize workflows:

```
{
  "workflowId": "uuid",
  "tenantId": "tenant-123",
  "configOverrides": {
    "temperature": 0.5,
    "max_iterations": 3
  },
  "disabledSteps": [3, 5],
  "modelPreferences": {
    "generator": "anthropic/claude-3-5-sonnet-20241022",
    "critic": "openai/o1"
  }
}
```

Parameter Schema

Each method has a JSON Schema for parameters:

```
{
  "type": "object",
  "properties": {
    "temperature": {
      "type": "number",
      "min": 0,
```

```
      "max": 2,
      "description": "Controls randomness"
    },
    "max_tokens": {
      "type": "integer",
      "description": "Maximum output length"
    }
  }
}
```

API Endpoints

Method Management

- `GET /api/admin/orchestration/methods` - List all methods
- `GET /api/admin/orchestration/methods/:code` - Get method details
- `PATCH /api/admin/orchestration/methods/:code` - Update method parameters

Workflow Management

- `GET /api/admin/orchestration/workflows` - List all workflows
- `GET /api/admin/orchestration/workflows/:code` - Get workflow with steps
- `POST /api/admin/orchestration/workflows/:code/customize` - Create tenant customization

Metrics

- `GET /api/admin/orchestration/metrics` - Aggregated metrics
- `GET /api/admin/orchestration/metrics/:workflowCode` - Workflow-specific metrics
- `GET /api/admin/orchestration/executions` - Recent executions

Stream Data Flow

Input Variables Available

Variable	Description	Source
`{{prompt}}`	Original user prompt	Request
`{{context}}`	Additional context	Request
`{{response}}`	Previous step output	Step N-1
`{{responses}}`	Multiple outputs	Parallel steps

Variable	Description	Source
`{{original_prompt}}`	Original prompt (unchanged)	Request
`{{feedback}}`	Critique output	Critic step
`{{challenge}}`	Challenge output	Challenger step

Output Structure

Each step produces:

```
{
  "response": "string",
  "tokens": 1234,
  "confidence": 0.85,
  "metadata": {}
}
```

For parallel execution with `outputMode: 'single'` (default):

```
{
  "response": "synthesized response string",
  "streamCount": 1,
  "outputMode": "single",
  "synthesisApplied": true,
  "modelsUsed": ["openai/o1", "claude-3-5-sonnet", "deepseek-reasoner"]
}
```

For parallel execution with `outputMode: 'all'`:

```
{
  "responses": [
    { "modelId": "openai/o1", "modelName": "o1", "response": "...", "confidence": 0.92, "lat
    { "modelId": "claude-3-5-sonnet", "modelName": "Claude 3.5 Sonnet", "response": "...", "
    { "modelId": "deepseek-reasoner", "modelName": "DeepSeek Reasoner", "response": "...", "
  ],
  "streamCount": 3,
  "outputMode": "all",
  "synthesisApplied": false,
  "modelsUsed": ["openai/o1", "claude-3-5-sonnet", "deepseek-reasoner"]
}
```

Multi-Model Parallel Execution

Overview

Any method can utilize **N models** simultaneously via the `parallelExecution` configuration. This enables: - **Consensus building** - Multiple perspectives on

the same problem - **Quality improvement** - Best-of-N selection - **Robustness** - Fallback if one model fails - **Diverse outputs** - Different approaches to creative tasks

Parallel Execution Configuration

```typescript
interface ParallelExecutionConfig {
  // Core settings
  enabled: boolean;
  mode: 'all' | 'race' | 'quorum';
  models: string[];

  // Synthesis
  synthesizeResults?: boolean;
  synthesisStrategy?: 'best_of' | 'merge' | 'vote' | 'weighted';
  weightByConfidence?: boolean;

  // AGI Dynamic Model Selection
  agiModelSelection?: boolean;
  minModels?: number;        // Default: 2
  maxModels?: number;        // Default: 5
  domainHints?: string[];

  // Failure handling
  timeoutMs?: number;
  quorumThreshold?: number; // For quorum mode: 0.5 = majority
  failureStrategy?: 'fail_fast' | 'continue' | 'fallback';

  // OUTPUT STREAM CONFIGURATION
  outputMode?: 'single' | 'all' | 'top_n' | 'threshold';
  outputTopN?: number;            // For top_n mode (default: 2)
  outputThreshold?: number;       // For threshold mode (default: 0.7)
  preserveModelAttribution?: boolean;
}
```

Execution Modes

Mode	Behavior	Use Case
`all`	Wait for all models to complete	Quality-critical tasks
`race`	Return first successful response	Latency-critical tasks
`quorum`	Wait for majority (threshold configurable)	Balanced approach

Synthesis Strategies

Strategy	Description
`best_of`	Select highest confidence response
`merge`	Combine all responses into one
`vote`	Majority answer wins
`weighted`	Weight by confidence scores

Output Stream Modes (Detailed)

Why Output Modes Matter

When a method uses 3 models, the question is: **how many streams should flow to the next step?**

- **Single stream** (default): Synthesize into one response for simple pipelines
- **All streams**: Pass all model outputs for the next step to compare/judge
- **Top N streams**: Only the best N by confidence
- **Threshold streams**: Only those above a quality bar

Mode Reference

`single` (Default)

```
3 Models    Synthesize    1 Stream    {{response}}
```

- **Use when**: Next step expects a single input
- **Output variable**: `{{response}}`
- **Example**: GENERATE → CRITIQUE (critic evaluates one response)

`all`

```
3 Models    3 Streams    {{responses}}[3]
```

- **Use when**: Next step needs to compare/synthesize multiple perspectives
- **Output variable**: `{{responses}}` array
- **Example**: 3x GENERATE → JUDGE_RESPONSES → pick best

`top_n`

```
3 Models    Sort by confidence    Top 2    {{responses}}[2]
```

- **Use when**: You want diversity but filtered by quality
- **Config**: `outputTopN: 2`
- **Output variable**: `{{responses}}` array

`threshold`

```
3 Models   Filter 80%   0-3 Streams   {{responses}}[0-3]
```

- **Use when**: Only high-quality responses should proceed
- **Config**: `outputThreshold: 0.8`
- **Output variable**: `{{responses}}` array (may be empty!)

Configuration Examples

Example 1: Multi-model critique with all perspectives

```
{
  stepName: 'Multi-Perspective Critique',
  method: 'CRITIQUE_RESPONSE',
  parallelExecution: {
    enabled: true,
    mode: 'all',
    models: ['openai/o1', 'claude-3-5-sonnet', 'deepseek-reasoner'],
    outputMode: 'all',                    // Pass all 3 critiques
    preserveModelAttribution: true        // Know which model said what
  }
}
// Next step receives {{responses}} with 3 critique objects
```

Example 2: Best-of-3 generation

```
{
  stepName: 'Generate with Best Selection',
  method: 'GENERATE_RESPONSE',
  parallelExecution: {
    enabled: true,
    mode: 'all',
    models: ['claude-3-5-sonnet', 'gpt-4o', 'gemini-pro'],
    outputMode: 'top_n',
    outputTopN: 1,                        // Only best response
    synthesizeResults: false              // Don't merge, just pick
  }
}
// Next step receives {{response}} (single best)
```

Example 3: Quality-filtered ensemble

```
{
  stepName: 'High-Confidence Ensemble',
  method: 'GENERATE_WITH_COT',
  parallelExecution: {
    enabled: true,
    agiModelSelection: true,              // AGI picks models
```

```
    minModels: 3,
    maxModels: 5,
    outputMode: 'threshold',
    outputThreshold: 0.85,                // Only 85% confidence
    preserveModelAttribution: true
  }
}
// Next step receives {{responses}} with only high-confidence outputs
```

Stream Flow Diagrams

Single Mode (Default)

```
   o1             Claude        DeepSeek




                Synthesize


            {{response}} = "..."


               Next Step
             (single input)
```

All Mode

```
   o1             Claude        DeepSeek
 conf: 0.92     conf: 0.88     conf: 0.85



{{responses}} = [
  { modelId: 'o1', response: '...', confidence: 0.92 },
  { modelId: 'claude', response: '...', confidence: 0.88 },
  { modelId: 'deepseek', response: '...', confidence: 0.85 }
]
```

```
   Next Step
(3 inputs)
JUDGE_RESPONSES
```

Top N Mode (N=2)

```
   o1             Claude         DeepSeek
 conf: 0.92     conf: 0.88     conf: 0.85
```

```
                                    (filtered out)
{{responses}} = [
  { modelId: 'o1', response: '...', confidence: 0.92 },
  { modelId: 'claude', response: '...', confidence: 0.88 }
]
```

```
   Next Step
(2 inputs)
```

Model Modes

Each model can run in a specialized mode for optimal performance:

Mode	Description	Best For
`standard`	Default execution	General tasks
`thinking`	Extended reasoning (o1, Claude thinking)	Complex logic
`deep_research`	In-depth research mode	Research tasks
`fast`	Speed-optimized (flash models)	Simple queries
`creative`	Higher temperature	Creative writing
`precise`	Low temperature, factual	Data extraction
`code`	Code-specialized	Programming
`vision`	Multimodal with vision	Image analysis
`long_context`	Extended context handling	Long documents

AGI Model Selection with Modes

When `agiModelSelection: true`, the system: 1. Analyzes the prompt/domain 2. Scores available models 3. Assigns optimal modes to each 4. Selects 2-5

models automatically

```
// AGI selection result
{
  selectedModels: [
    { modelId: 'openai/o1', mode: 'thinking' },
    { modelId: 'claude-3-5-sonnet', mode: 'standard' },
    { modelId: 'deepseek-reasoner', mode: 'deep_research' }
  ],
  reasoning: 'Selected 3 models with reasoning modes for complex analysis task',
  domainDetected: 'science',
  executionStrategy: 'parallel'
}
```

Proficiency System

Overview

The proficiency system is the **bridge between prompts and model selection**. It enables domain-aware orchestration by scoring both **domains** and **models** across 8 dimensions.

8 Proficiency Dimensions (1-10 scale)

Dimension	Description	High Score Means
`reasoning_depth`	Depth of logical reasoning required	Complex deduction, multi-step logic
`mathematical_quantitative`	Mathematical/quantitative analysis	Calculations, statistics, proofs
`code_generation`	Code writing/debugging capability	Programming tasks
`creative_generative`	Creative/generative content	Stories, art, brainstorming
`research_synthesis`	Research and synthesis ability	Literature review, analysis
`factual_recall_precision`	Factual accuracy requirements	Facts, definitions, dates
`multi_step_problem_solving`	Complex problem decomposition	Breaking down hard problems
`domain_terminology_handling`	Domain-specific jargon handling	Technical vocabulary

How Proficiencies Flow Through the System

```
USER PROMPT: "Derive the Navier-Stokes equations for incompressible flow"



STEP 1: DOMAIN DETECTION

Matched: Science → Physics → Fluid Dynamics
Confidence: 0.94

Detected Keywords: "Navier-Stokes", "equations", "incompressible", "derive"



STEP 2: PROFICIENCY EXTRACTION

Each level has proficiency scores that get MERGED:

Field (Science):              Domain (Physics):         Subspecialty (Fluid):
  reasoning: 8                  reasoning: 9              reasoning: 9
  math: 7                       math: 10                  math: 10
  code: 3                       code: 4                   code: 5
  creative: 4                   creative: 3               creative: 2
  research: 7                   research: 8               research: 7
  factual: 8                    factual: 9                factual: 8
  multi_step: 7                 multi_step: 9             multi_step: 10
  terminology: 6                terminology: 8            terminology: 9

MERGED PROFICIENCIES (weighted by specificity):
{
  reasoning_depth: 9,
  mathematical_quantitative: 10,
  code_generation: 5,
  creative_generative: 2,
  research_synthesis: 7,
  factual_recall_precision: 8,
  multi_step_problem_solving: 10,
  domain_terminology_handling: 9
}
```

```
STEP 3: ORCHESTRATION MODE SELECTION

Proficiency-based rules:
  reasoning_depth >= 9 AND multi_step >= 9 → extended_thinking
  code_generation >= 8 → coding
  creative_generative >= 8 → creative
  research_synthesis >= 8 → research
  mathematical_quantitative >= 8 → analysis

Selected: extended_thinking
Reason: "Complex reasoning required based on domain proficiencies"



STEP 4: MODEL MATCHING

Each model has proficiency scores. Match against domain requirements:

Model: OpenAI o1                    Match Score: 94%
  reasoning: 10 (need 9)   +1
  math: 9 (need 10) -1
  multi_step: 10 (need 10)
  Strengths: [reasoning, multi_step, math]

Model: Claude 3.5 Sonnet            Match Score: 87%
  reasoning: 9 (need 9)
  math: 8 (need 10) -2
  Strengths: [reasoning, research, terminology]

Model: DeepSeek Reasoner            Match Score: 91%
  reasoning: 10 (need 9)   +1
  math: 10 (need 10)
  Strengths: [math, reasoning, code]

SELECTED: o1 (primary), DeepSeek (fallback), Claude (fallback)



STEP 5: EXECUTION

parallelExecution: {
  enabled: true,
  models: ['openai/o1', 'deepseek-reasoner'],  // Both strong in math+reason
```

```
    mode: 'all',
    outputMode: 'top_n',
    outputTopN: 1,  // Pick best
    synthesisStrategy: 'best_of'
  }
```

Proficiency Types in the Hierarchy

```
Field (Top Level)
  field_proficiencies: ProficiencyScores

      Domain (Middle Level)
          domain_proficiencies: ProficiencyScores

              Subspecialty (Leaf Level)
                  subspecialty_proficiencies: ProficiencyScores
```

Model Proficiency Matching

```typescript
interface ModelProficiencyMatch {
  model_id: string;
  provider: string;
  model_name: string;
  match_score: number;           // 0-100 overall match
  dimension_scores: Record<ProficiencyDimension, number>;
  strengths: ProficiencyDimension[];
  weaknesses: ProficiencyDimension[];
  recommended: boolean;
  ranking: number;
}
```

Proficiency → Mode Decision Table

Proficiency Condition	Orchestration Mode	Reason
`reasoning_depth >= 9 AND multi_step >= 9`	`extended_thinking`	Complex logical reasoning
`code_generation >= 8`	`coding`	Programming task
`creative_generative >= 8`	`creative`	Creative writing
`research_synthesis >= 8`	`research`	Research/analysis
`mathematical_quantitative >= 8`	`analysis`	Quantitative work

Proficiency Condition	Orchestration Mode	Reason
High `factual_recall` + sensitive topic	`self_consistency`	Accuracy critical
Default	`thinking`	Standard reasoning

Proficiency → Model Strengths Mapping

Model	Top Proficiencies	Best For
OpenAI o1	reasoning_depth (10), multi_step (10)	Complex reasoning
Claude 3.5 Sonnet	reasoning (9), research (9), terminology (9)	Research, analysis
DeepSeek Reasoner	math (10), reasoning (10), code (8)	Math, logic, code
GPT-4o	creative (8), research (8), factual (8)	General, creative
Gemini Pro	math (8), code (8), research (8)	Technical analysis
Claude Haiku	factual (7), terminology (7)	Quick answers

Example: Proficiency-Driven Workflow

```
// 1. Detect domain and get proficiencies
const detection = await domainTaxonomyService.detectDomain(prompt);
// Returns: { merged_proficiencies: { reasoning_depth: 9, math: 10, ... } }

// 2. Determine orchestration mode from proficiencies
const mode = determineOrchestrationMode(detection.merged_proficiencies);
// Returns: 'extended_thinking' (because reasoning >= 9 and multi_step >= 9)

// 3. Match models to proficiencies
const matches = await domainTaxonomyService.getMatchingModels(
  detection.merged_proficiencies,
  { max_models: 3, min_match_score: 80 }
);
// Returns: [{ model_id: 'o1', match_score: 94 }, { model_id: 'deepseek', match_score: 91 }]

// 4. Execute with matched models
const result = await orchestrationService.executeWorkflow({
  workflowCode: 'EXTENDED_THINKING_DUAL',
  parallelExecution: {
    enabled: true,
    models: matches.map(m => m.model_id),
    outputMode: 'top_n',
    outputTopN: 1
```

```
  }
});
```

Admin: Viewing Proficiencies

Admin Dashboard → Orchestration → Methods → Parallel & Streams tab

Shows: - Domain proficiency requirements for the task - Model match scores - Which dimensions drove model selection - Output stream configuration

---

AGI Brain + Workflow Integration

The AGI Brain Planner can **select and configure workflows** to solve problems. Users can either let the AGI choose the optimal workflow or specify their preferences.

User Choices for Workflows

```
interface GeneratePlanRequest {
  prompt: string;
  tenantId: string;
  userId: string;

  // ... other options ...

  // Workflow Selection - User choices
  preferredWorkflow?: string;           // User-selected workflow code
  workflowParameterOverrides?: Record<string, unknown>;  // User parameter tweaks
  allowAgiWorkflowSelection?: boolean;  // Let AGI pick workflow (default: true)
  excludeWorkflows?: string[];          // Workflows to exclude from selection
}
```

How AGI Selects Workflows

```
  PROMPT: "Write a comprehensive analysis of renewable energy trends"



  STEP 1: Check User Preference

  Did user specify preferredWorkflow?
    YES → Use that workflow with user's parameter overrides
    NO  → Continue to AGI selection
```

```
STEP 2: AGI Workflow Selection

orchestrationPatternsService.selectPattern({
  prompt: "Write a comprehensive analysis...",
  taskType: "research",
  complexity: "complex",
  qualityPriority: 0.9,
  costSensitive: false,
  excludePatterns: user.excludeWorkflows
})

Scores 49 available workflows against problem characteristics




STEP 3: Selected Workflow

selectedWorkflow: {
  workflowCode: 'RESEARCH_SYNTHESIS_MULTI',
  workflowName: 'Multi-Model Research Synthesis',
  selectionReason: 'Matches research task, high quality priority',
  selectionConfidence: 0.89,
  selectionMethod: 'auto'
}

alternatives: [
  { workflowCode: 'CHAIN_OF_THOUGHT', matchScore: 0.82 },
  { workflowCode: 'SELF_CONSISTENCY', matchScore: 0.78 }
]




STEP 4: Workflow Steps with Parameters

workflowSteps: [
  {
    methodCode: 'DECOMPOSE_PROBLEM',
    parameterOverrides: { max_subproblems: 5 }
  },
  {
```

```
      methodCode: 'GENERATE_RESPONSE',
      isParallel: true,
      parallelConfig: {
        models: ['claude-3-5-sonnet', 'gpt-4o', 'gemini-pro'],
        outputMode: 'all'  // 3 streams to next step
      }
    },
    {
      methodCode: 'SYNTHESIZE_RESPONSES',
      parameterOverrides: { combination_strategy: 'best_parts' }
    }
  ]

  workflowConfig: {
    ...defaultConfig,
    ...userParameterOverrides  // User's tweaks merged in
  }
```

Example: User Specifies Workflow + Parameters

```
// User explicitly chooses a workflow and tweaks parameters
const plan = await agiBrainPlannerService.generatePlan({
  prompt: "Debug this React component that crashes on mount",
  tenantId: "tenant-123",
  userId: "user-456",

  // User choices
  preferredWorkflow: 'SELF_REFINE_LOOP',  // User picks this workflow
  workflowParameterOverrides: {
    max_iterations: 5,                     // More refinement rounds
    temperature: 0.2,                      // More precise
    refinement_focus: 'accuracy'
  }
});

// Result includes:
// - selectedWorkflow.selectionMethod = 'user'
// - workflowConfig merged with user overrides
// - workflowSteps configured with user parameters
```

Example: AGI Auto-Selects Workflow

```
// Let AGI choose the best workflow
const plan = await agiBrainPlannerService.generatePlan({
  prompt: "Compare the economic policies of three countries",
```

```
  tenantId: "tenant-123",
  userId: "user-456",

  allowAgiWorkflowSelection: true,  // default
  excludeWorkflows: ['FAST_SIMPLE']  // User says: not this one
});

// AGI analyzes prompt and selects:
// - selectedWorkflow.workflowCode = 'MULTI_AGENT_DEBATE'
// - selectedWorkflow.selectionMethod = 'domain_match'
// - selectedWorkflow.selectionReason = 'Comparison task benefits from debate'
// - alternatives = [other matching workflows]
```

Plan Output with Workflow

```
interface AGIBrainPlan {
  // ... existing fields ...

  // Workflow Integration
  selectedWorkflow?: {
    workflowId: string;
    workflowCode: string;
    workflowName: string;
    description: string;
    category: string;
    selectionReason: string;
    selectionConfidence: number;
    selectionMethod: 'auto' | 'user' | 'domain_match';
  };

  workflowSteps?: Array<{
    bindingId: string;
    stepOrder: number;
    methodCode: string;
    methodName: string;
    parameterOverrides: Record<string, unknown>;
    dependsOn: string[];
    isParallel: boolean;
    parallelConfig?: {
      models: string[];
      outputMode: 'single' | 'all' | 'top_n' | 'threshold';
    };
  }>;

  workflowConfig?: Record<string, unknown>;
```

```
  alternativeWorkflows?: Array<{
    workflowCode: string;
    workflowName: string;
    matchScore: number;
    reason: string;
  }>;
}
```

Specialty Categories (Domain Expertise)

> **Full Documentation**: See SPECIALTY-RANKING.md for complete details on the specialty ranking system, AI-powered research, admin controls, and database schema.

In addition to the 8 proficiency dimensions, models are ranked across **20 specialty categories** representing domain-specific expertise:

Specialty Categories

Category	Icon	Description
`reasoning`		Reasoning & Logic
`coding`		Code Generation
`math`		Mathematics
`creative`		Creative Writing
`analysis`		Data Analysis
`research`		Research & Synthesis
`legal`		Legal & Compliance
`medical`		Medical & Healthcare
`finance`		Finance & Trading
`science`		Scientific
`debugging`		Debugging & QA
`architecture`		System Architecture
`security`		Security
`vision`		Vision & Images
`audio`		Audio & Speech
`conversation`		Conversational
`instruction`		Instruction Following
`speed`		Low Latency
`accuracy`		High Accuracy
`safety`		Safety & Alignment

Two-Layer Proficiency System

```
LAYER 1: TASK PROFICIENCY DIMENSIONS (8)
From domain taxonomy - "What capabilities does this task need?"

  reasoning_depth              Multi-step logical thinking
  mathematical_quantitative    Calculations, proofs, statistics
  code_generation              Programming tasks
  creative_generative          Stories, art, ideas
  research_synthesis           Literature review, analysis
  factual_recall_precision     Facts, definitions, accuracy
  multi_step_problem_solving   Breaking down complex problems
  domain_terminology_handling  Technical vocabulary


                                    Drives


LAYER 2: SPECIALTY CATEGORIES (20)
Per-model rankings - "How good is each model in each specialty?"

Domain Expertise:                 Performance Attributes:
    medical                         speed
    legal                           accuracy
    finance                         safety
    science                         instruction
    security
    architecture                Modalities:
                                       vision
Task Capabilities:                     audio
    reasoning
    coding
    math
    creative
    analysis
    research
    debugging
    conversation
```

How Both Layers Work Together

```
PROMPT: "Analyze this ECG reading and suggest treatment options"
```

```
STEP 1: Domain Detection
→ Field: Medicine → Domain: Cardiology → Subspecialty: Diagnostics
→ Confidence: 0.91



STEP 2: Extract TASK PROFICIENCY Requirements

From domain taxonomy:
{
  reasoning_depth: 8,              // Diagnostic reasoning
  mathematical_quantitative: 5,    // Some measurements
  factual_recall_precision: 9,     // Medical accuracy critical
  research_synthesis: 7,           // Treatment guidelines
  domain_terminology_handling: 10  // Medical jargon
}



STEP 3: Query SPECIALTY RANKINGS for Models

Required specialties: medical + accuracy + safety + research

Model Specialty Scores:

 Model                    Medical   Accuracy   Safety   Research

 Claude 3.5 Sonnet      92 (A)     91 (A)     95 (S)   90 (A)
 GPT-4o                 88 (A)     89 (A)     90 (A)   87 (A)
 DeepSeek Medical*      95 (S)     85 (B)     88 (A)   82 (B)
 Gemini Pro             84 (B)     86 (B)     89 (A)   88 (A)


* Self-hosted domain-specific model



STEP 4: Combined Scoring

Final Score = TaskProficiencyMatch × SpecialtyScore × SafetyWeight

Claude 3.5 Sonnet: 0.88 × 92 × 1.2 = 97.2 ← SELECTED (primary)
DeepSeek Medical:  0.82 × 95 × 1.0 = 77.9 ← SELECTED (fallback)
```

```
  GPT-4o:            0.85 × 88 × 1.1 = 82.3 ← SELECTED (fallback)



  STEP 5: Execution with Multi-Model

  parallelExecution: {
    enabled: true,
    models: ['claude-3-5-sonnet', 'deepseek-medical', 'gpt-4o'],
    outputMode: 'threshold',
    outputThreshold: 0.85,  // Only high-confidence medical advice
    synthesisStrategy: 'weighted'  // Weight by specialty scores
  }
```

Specialty Ranking Structure

```typescript
interface SpecialtyRanking {
  rankingId: string;
  modelId: string;
  provider: string;
  specialty: SpecialtyCategory;     // 'medical', 'legal', 'coding', etc.
  proficiencyScore: number;          // 0-100 overall score
  benchmarkScore: number;            // 0-100 from published benchmarks
  communityScore: number;            // 0-100 from community reviews
  internalScore: number;             // 0-100 from internal usage data
  rank: number;                      // Global rank for this specialty
  percentile: number;                // e.g., top 10%
  tier: 'S' | 'A' | 'B' | 'C' | 'D' | 'F';  // Quality tier
  confidence: number;                // 0-1 confidence in assessment
  trend: 'improving' | 'stable' | 'declining';
  adminOverride?: number;            // Admin can lock a score
  isLocked: boolean;
}
```

Tier System

Tier	Score Range	Description
S	95-100	Elite - Best-in-class for this specialty
A	85-94	Excellent - Highly recommended
B	75-84	Good - Solid performance
C	65-74	Average - Acceptable
D	50-64	Below Average - Use with caution
F	0-49	Poor - Not recommended

Example: Model Specialty Profiles

Claude 3.5 Sonnet

```
reasoning:     94 (S)
coding:        95 (S)
math:          88 (A)
creative:      92 (A)
medical:       92 (A)
legal:         89 (A)
security:      91 (A)
safety:        95 (S)
```

OpenAI o1

```
reasoning:     98 (S)
coding:        90 (A)
math:          96 (S)
creative:      75 (B)
medical:       85 (B)
legal:         88 (A)
security:      89 (A)
safety:        92 (A)
```

DeepSeek Coder

```
reasoning:     85 (B)
coding:        96 (S)
math:          92 (A)
creative:      65 (C)
debugging:     94 (S)
architecture: 88 (A)
speed:         90 (A)
```

AI-Powered Research

The specialty rankings are maintained through **automated AI research**:

```
// Research model proficiency across all specialties
const result = await specialtyRankingService.researchModelProficiency('anthropic/claude-3-5-

// Research all models for a specific specialty
const result = await specialtyRankingService.researchSpecialtyRankings('medical');
```

Research sources include: - Published benchmarks (MMLU, HumanEval, MATH, etc.) - Community reviews and feedback - Internal usage data and quality scores - Domain-specific evaluations

Admin Controls

Admins can: - **Override scores**: Lock a model's specialty score - **View leaderboards**: See top models per specialty - **Trigger research**: Refresh rankings from latest data - **Configure weights**: Adjust benchmark vs community vs internal weighting

---

SEED-DATA-SYSTEM

RADIANT AI Registry Seed Data System

Technical Documentation

Version: 4.18.1 | Last Updated: December 2024

Overview

The RADIANT Seed Data System manages versioned AI provider and model configurations that are used to populate the AI Registry during fresh installations. Seed data is stored separately from packages, can be versioned independently, and is selectable when building deployment packages.

Architecture

```
                    SEED DATA ARCHITECTURE


config/seeds/
  registry.json          # Index of all seed versions
  v1/                    # Seed data version 1.0.0
     manifest.json      # Version metadata and stats
     providers.json     # 21 external providers
     external-models.json  # 50+ external models
     self-hosted-models.json  # 38 self-hosted models
     services.json      # 5 orchestration services
  v2/                    # Future seed versions...

Build Time:

 build-package.sh    Select seed version    Include in package
 --seed-version 1
```

```
Deploy Time (INSTALL only):

  DeploymentService   Read seeds from package    INSERT to database
  .executeInstall()
```

Critical Rules

Rule 1: NO HARDCODING IN DEPLOYER APP

The Swift Deployer app **MUST NOT** contain hardcoded lists of providers or models:

```
//  WRONG - Never do this
let providers = ["openai", "anthropic", "google", ...]

//  CORRECT - Fetch from Radiant API after deployment
let providers = try await radiantAPI.fetchProviders()
```

Rule 2: INSTALLER SEEDS, UPDATER PRESERVES

Mode	Seed Behavior
INSTALL	Seeds database with complete provider/model list
UPDATE	NEVER touches AI Registry - preserves admin customizations
ROLLBACK	Restores from snapshot - does not re-seed

Rule 3: ADMIN CONTROLS ALL

Everything in seed data is **editable by the administrator** post-deployment: - Enable/disable providers and models - Change pricing markup - Add new providers/models - Delete providers/models

---

Seed Data Structure

manifest.json

```
{
  "version": "1.0.0",
  "name": "RADIANT AI Registry Seed Data",
  "description": "Complete provider and model seed data for fresh installations",
  "createdAt": "2024-12-25T00:00:00Z",
  "updatedAt": "2024-12-25T00:00:00Z",
```

```
  "compatibility": {
    "minRadiantVersion": "4.16.0",
    "maxRadiantVersion": "5.0.0"
  },
  "files": {
    "providers": "providers.json",
    "externalModels": "external-models.json",
    "selfHostedModels": "self-hosted-models.json",
    "services": "services.json"
  },
  "stats": {
    "externalProviders": 21,
    "externalModels": 50,
    "selfHostedModels": 38,
    "services": 5
  },
  "pricing": {
    "externalMarkup": 1.40,
    "selfHostedMarkup": 1.75
  }
}
```

providers.json

Each provider includes:

Field	Description
`id`	Unique identifier
`name`	Internal name
`displayName`	Human-readable name
`category`	Provider category (text_generation, image_generation, etc.)
`apiBaseUrl`	API endpoint
`authType`	Authentication type (bearer, api_key, iam)
`secretName`	AWS Secrets Manager path for API key
`features`	Supported features (streaming, vision, etc.)
`compliance`	Compliance certifications (SOC2, GDPR, HIPAA)
`rateLimit`	Rate limiting configuration

external-models.json

Each model includes:

Field	Description
`id`	Unique model identifier
`providerId`	Reference to provider
`modelId`	Provider's model ID
`litellmId`	LiteLLM routing ID
`category`	Model category
`capabilities`	Model capabilities
`contextWindow`	Max input tokens
`maxOutput`	Max output tokens
`pricing`	Cost per 1K tokens + markup
`minTier`	Minimum tier required

self-hosted-models.json

Each self-hosted model includes:

Field	Description
`id`	Unique model identifier
`instanceType`	SageMaker instance type
`thermal`	Thermal management config (COLD/WARM/HOT)
`license`	Open-source license
`pricing`	Hourly rate + per-unit pricing
`minTier`	Minimum tier required (typically 3+)

Building Packages with Seed Data

List Available Seed Versions

```
./tools/scripts/build-package.sh --list-seeds
```

Output:

```
Available Seed Data Versions:
  v1.0.0 - 21 providers, 50 external models, 38 self-hosted models
```

Build with Specific Seed Version

```
# Use default (latest) seed version
./tools/scripts/build-package.sh

# Use specific seed version
./tools/scripts/build-package.sh --seed-version 1
```

Package Manifest with Seed Data

The generated package manifest includes seed data information:

```
{
  "schemaVersion": "2.1",
  "package": {
    "version": "4.18.1"
  },
  "seedData": {
    "version": "1.0.0",
    "hash": "abc123...",
    "externalProviders": 21,
    "externalModels": 50,
    "selfHostedModels": 38,
    "services": 5
  },
  "installBehavior": {
    "seedAIRegistry": true
  },
  "updateBehavior": {
    "seedAIRegistry": false
  }
}
```

Seed Data Categories

External Providers (21)

Category	Providers
Text Generation	OpenAI, Anthropic, Google, xAI, DeepSeek, Mistral, Cohere
Image Generation	OpenAI Images, Stability AI, FLUX
Video Generation	Runway, Luma AI
Audio	ElevenLabs, OpenAI Audio
Embeddings	OpenAI Embeddings, Voyage AI
Search	Perplexity
3D Generation	Meshy
Self-Hosted	SageMaker (internal)

External Models (50+)

Category	Example Models
Text	GPT-4o, Claude Sonnet 4, Gemini 2.0, Grok 3, DeepSeek R1
Reasoning	O1, O3 Mini, DeepSeek Reasoner
Code	Codestral
Image	DALL-E 3, Stable Diffusion 3, FLUX Pro
Video	Gen-3 Alpha, Ray 2
Audio	Whisper, TTS-1, Multilingual V2

Self-Hosted Models (38)

Category	Models
Vision Classification	EfficientNet, Swin Transformer, CLIP
Object Detection	YOLOv8 (Nano/Small/Medium/XLarge), Grounding DINO
Segmentation	SAM, SAM 2, MobileSAM
Speech	Whisper Large V3, Parakeet TDT
Scientific	AlphaFold 2, ESM-2
Medical	nnU-Net, MedSAM
Geospatial	Prithvi 100M/600M
3D	Nerfstudio
LLM	Mistral 7B, Llama 3 70B

Pricing Structure

External Providers

Default markup: **40% (1.40x)**

Example: GPT-4o - Provider cost: $0.0025/1K input, $0.01/1K output - Tenant cost: $0.0035/1K input, $0.014/1K output

Self-Hosted Models

Default markup: **75% (1.75x)**

Example: YOLOv8 Medium - Infrastructure cost: ~$2.47/hour + $0.005/image - Tenant cost: ~$4.32/hour + $0.00875/image

---

Creating New Seed Versions

1. Create Version Directory

```
mkdir config/seeds/v2
```

2. Create Required Files

- `manifest.json` - Version metadata
- `providers.json` - Provider definitions
- `external-models.json` - External model definitions
- `self-hosted-models.json` - Self-hosted model definitions
- `services.json` - Service definitions

3. Update Registry

Add new version to `config/seeds/registry.json`:

```
{
  "versions": [
    {
      "version": "2.0.0",
      "directory": "v2",
      "releaseDate": "2025-01-15",
      "status": "stable",
      "changelog": "Added new providers and models..."
    },
    // ... existing versions
  ]
}
```

4. Test Build

```
./tools/scripts/build-package.sh --seed-version 2
```

Database Seeding

During fresh installation, the DeploymentService generates SQL migrations from seed data:

```
-- Only runs if providers table is empty
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM providers LIMIT 1) THEN
        -- Insert providers
        INSERT INTO providers (...) VALUES (...);

        -- Insert external models
        INSERT INTO models (...) VALUES (...);

        -- Insert self-hosted models
        INSERT INTO self_hosted_models (...) VALUES (...);
```

```
    END IF;
END $$;
```

Key behaviors: - Uses `ON CONFLICT DO NOTHING` to preserve admin changes - Only runs on fresh install (empty database) - Logs completion with model counts

---

Swift Service API

SeedDataService

```swift
actor SeedDataService {
    /// List available seed versions
    func listAvailableSeedVersions() async throws -> [SeedDataInfo]

    /// Load complete seed data for a version
    func loadSeedData(version: String) async throws -> SeedData

    /// Generate SQL migration from seed data
    func generateSeedMigration(seedData: SeedData) -> String
}
```

Usage in DeploymentService

```swift
func executeInstall(...) async throws -> DeploymentExecutionResult {
    // Load seed data from package
    let seedData = try await seedDataService.loadSeedData(
        version: package.manifest.seedData?.version ?? "1.0.0"
    )

    // Generate and run seed migration
    let seedSQL = seedDataService.generateSeedMigration(seedData: seedData)
    try await runMigration(sql: seedSQL)
}
```

Related Documentation

- Deployer Architecture - Deployment modes and package management
- Deployer Admin Guide - User-facing deployment documentation
- API Reference - Provider and model API endpoints

REVENUE-ANALYTICS

Revenue Analytics System

Version: 4.18.3
Last Updated: 2024-12-28

Overview

The Revenue Analytics system tracks gross revenue, cost of goods sold (COGS), and gross profit across all RADIANT products. It provides visibility into subscription billing, AI provider markup, self-hosted model revenue, and associated AWS/infrastructure costs.

Important: This system tracks **gross revenue and COGS only**. Marketing, sales, G&A, and other operating expenses must be subtracted separately in your accounting system to calculate net profit.

Admin Dashboard Location

Path: Admin Dashboard → Revenue Analytics
URL: `/revenue`

Revenue Sources

Source	Description	Example
`subscription`	Monthly/annual subscription fees	Tier 3 Pro @ $99/month
`credit_purchase`	One-time credit purchases	10,000 credits @ $50
`ai_markup_external`	Markup on external AI provider usage	OpenAI, Anthropic, etc. (typically 20-35% markup)
`ai_markup_self_hosted`	Markup on self-hosted model usage	SageMaker models (typically 75% markup on AWS cost)
`overage`	Usage beyond subscription limits	Additional API calls beyond tier limit
`storage`	Storage fees	User file storage, vector embeddings
`other`	Miscellaneous revenue	Custom integrations, support fees

Cost Categories (COGS)

Category	Description	AWS Services
`aws_compute`	Compute infrastructure	EC2, SageMaker, Lambda
`aws_storage`	Storage services	S3, EBS, EFS
`aws_network`	Network and data transfer	Data Transfer, API Gateway, CloudFront
`aws_database`	Database services	Aurora PostgreSQL, DynamoDB
`external_ai`	External AI provider costs	OpenAI API, Anthropic API, etc.
`infrastructure`	Other cloud costs	Secrets Manager, CloudWatch, etc.
`platform_fees`	Payment processing	Stripe fees (~2.9% + $0.30)

Dashboard Features

Summary Cards

- **Gross Revenue**: Total revenue from all sources
- **Total COGS**: Sum of all cost categories
- **Gross Profit**: Revenue minus COGS
- **Gross Margin**: (Profit / Revenue) × 100%

Time Period Selection

Period	Description
7d	Last 7 days
30d	Last 30 days (default)
90d	Last 90 days
YTD	Year to date
12m	Last 12 months

Tabs

1. **Revenue Breakdown**: Revenue by source with visual progress bars
2. **Cost Breakdown**: AWS costs and external provider costs
3. **By Model**: Per-model revenue with provider cost vs customer charge
4. **By Tenant**: Top tenants by total revenue

Export Formats

Available Formats

Format	File Extension	Use Case
CSV	`.csv`	Summary for spreadsheets (Excel, Google Sheets)
JSON	`.json`	Full details for custom integrations
QuickBooks IIF	`.iif`	Direct import to QuickBooks Desktop
Xero CSV	`.csv`	Import to Xero accounting
Sage CSV	`.csv`	Import to Sage accounting

Export Contents

CSV Summary Export:

```
Period Start,2024-12-01
Period End,2024-12-28

REVENUE,
Subscription Revenue,15000.00
Credit Purchase Revenue,2500.00
AI Markup (External),8750.00
AI Markup (Self-Hosted),3200.00
...
TOTAL GROSS REVENUE,29450.00

COSTS (COGS),
AWS Compute,4500.00
External AI Providers,7000.00
...
TOTAL COST,12500.00

GROSS PROFIT,16950.00
GROSS MARGIN,57.5%
```

QuickBooks IIF Export: - Creates General Journal entries - Revenue accounts: Subscription Revenue, Credit Sales Revenue, AI Markup Revenue - Expense accounts: AWS Compute Expense, External AI Provider Expense - Includes CLASS for categorization

---

API Endpoints

GET /api/admin/revenue/dashboard

Returns the revenue dashboard data.

Query Parameters: | Parameter | Type | Required | Description | |———–|——|———-|————-| | periodStart | ISO Date | Yes | Start of period | | periodEnd | ISO Date | Yes | End of period | | period | string | Yes | `day`, `week`, `month`, `quarter`, `year` | | tenantId | UUID | No | Filter to specific tenant |

Response:

```
{
  "summary": {
    "periodStart": "2024-12-01T00:00:00Z",
    "periodEnd": "2024-12-28T23:59:59Z",
    "subscriptionRevenue": 15000.00,
    "creditPurchaseRevenue": 2500.00,
    "aiMarkupExternalRevenue": 8750.00,
    "aiMarkupSelfHostedRevenue": 3200.00,
    "overageRevenue": 0,
    "storageRevenue": 0,
    "otherRevenue": 0,
    "totalGrossRevenue": 29450.00,
    "awsComputeCost": 4500.00,
    "awsStorageCost": 500.00,
    "awsNetworkCost": 200.00,
    "awsDatabaseCost": 800.00,
    "externalAiCost": 7000.00,
    "infrastructureCost": 300.00,
    "platformFeesCost": 850.00,
    "totalCost": 14150.00,
    "grossProfit": 15300.00,
    "grossMargin": 51.95
  },
  "previousPeriodSummary": { ... },
  "trends": [
    { "date": "2024-12-01", "grossRevenue": 1050.00, "totalCost": 450.00, ... }
  ],
  "byTenant": [
    { "tenantId": "uuid", "tenantName": "Acme Corp", "totalRevenue": 5000.00, ... }
  ],
  "byModel": [
    { "modelId": "gpt-4o", "hostingType": "external", "providerCost": 500.00, "customerCharg
  ],
  "revenueChange": 12.5,
  "profitChange": 15.2,
  "marginChange": 2.1
}
```

POST /api/admin/revenue/export

Exports revenue data in the specified format.

Request Body:

```
{
  "format": "quickbooks_iif",
  "periodStart": "2024-12-01T00:00:00Z",
  "periodEnd": "2024-12-28T23:59:59Z",
  "includeDetails": false,
  "tenantId": null
}
```

Response:

```
{
  "filename": "revenue_2024-12-01_2024-12-28.iif",
  "mimeType": "text/plain",
  "data": "base64-encoded-file-content",
  "recordCount": 14,
  "periodStart": "2024-12-01T00:00:00Z",
  "periodEnd": "2024-12-28T23:59:59Z"
}
```

Database Schema

revenue_entries

Individual revenue events.

Column	Type	Description
id	UUID	Primary key
tenant_id	UUID	FK to tenants
source	VARCHAR(30)	Revenue source type
amount	DECIMAL(15,4)	Revenue amount
currency	VARCHAR(3)	Currency code (default: USD)
description	TEXT	Description
reference_id	VARCHAR(255)	Related subscription/transaction ID
reference_type	VARCHAR(50)	Type of reference
product	VARCHAR(20)	`radiant`, `think_tank`, or `combined`
model_id	VARCHAR(100)	For AI markup revenue
period_start	TIMESTAMPTZ	Period this revenue applies to
period_end	TIMESTAMPTZ	Period end

cost_entries

Infrastructure and provider costs.

Column	Type	Description
id	UUID	Primary key
tenant_id	UUID	FK to tenants (NULL for shared infra)
category	VARCHAR(30)	Cost category
amount	DECIMAL(15,4)	Cost amount
aws_service_name	VARCHAR(100)	e.g., 'SageMaker', 'Aurora'
resource_id	VARCHAR(255)	AWS resource identifier
provider_id	VARCHAR(50)	For external AI costs
period_start	TIMESTAMPTZ	Period start
period_end	TIMESTAMPTZ	Period end

revenue_daily_aggregates

Pre-computed daily summaries for fast queries.

Column	Type	Description
aggregate_date	DATE	The date
tenant_id	UUID	NULL for platform totals
subscription_revenue	DECIMAL	Daily subscription revenue
ai_markup_external_revenue	DECIMAL	Daily external AI markup
ai_markup_self_hosted_revenue	DECIMAL	Daily self-hosted markup
total_gross_revenue	DECIMAL	Total for the day
total_cost	DECIMAL	Total COGS for the day
gross_profit	DECIMAL	Revenue - Cost
gross_margin	DECIMAL	Percentage margin

model_revenue_tracking

Per-model revenue breakdown for markup analysis.

Column	Type	Description
tracking_date	DATE	The date
model_id	VARCHAR(100)	Model identifier
hosting_type	VARCHAR(20)	`external` or `self_hosted`
provider_cost	DECIMAL	What we pay
customer_charge	DECIMAL	What customer pays
markup	DECIMAL	customer_charge - provider_cost
markup_percent	DECIMAL	Percentage markup
request_count	INTEGER	Number of requests

Markup Calculations

External AI Providers

Default markup: **20-35%** on provider cost

```
Customer Price = Provider Cost × (1 + Markup Rate)
Markup Revenue = Customer Price - Provider Cost
```

Self-Hosted Models

Default markup: **75%** on AWS SageMaker cost

```
Hourly Customer Rate = AWS Hourly Cost × 1.75
Per-Request Rate = AWS Cost + (AWS Cost × 0.75)
```

Accounting Integration Notes

QuickBooks

- Import the `.iif` file via File → Utilities → Import → IIF Files
- Creates General Journal entries with appropriate accounts
- Requires accounts to exist: Subscription Revenue, AWS Compute Expense, etc.

Xero

- Import via Invoices → Import
- Maps to account codes (4000-series for revenue, 5000-series for expenses)

Sage

- Import via Transactions → Import
- Uses nominal codes matching Sage's structure

Permissions

Revenue Analytics is visible only to: - **Platform Admins**: Full access to all tenant data - **Tenant Admins**: Access to their tenant's revenue only (filtered view)

Related Documentation

- Billing & Credits
- Cost Analytics

- Model Pricing
- Unified Model Registry

SAAS-METRICS-DASHBOARD

SaaS Metrics Dashboard

Version: 4.18.3
Last Updated: 2024-12-28

Overview

The SaaS Metrics Dashboard provides comprehensive visibility into key business metrics including revenue, costs, customer health, and model profitability. It integrates data from the Revenue Analytics system and Cost Analytics to present a unified view of business performance.

Admin Dashboard Location

Path: Admin Dashboard → SaaS Metrics
URL: `/saas-metrics`

Key Metrics

Revenue Metrics

Metric	Description	Formula
MRR	Monthly Recurring Revenue	Sum of all monthly subscription fees
ARR	Annual Recurring Revenue	MRR × 12
Total Revenue	All revenue in period	Subscriptions + Credits + AI Markup + Storage
ARPU	Average Revenue Per User	Total Revenue / Active Customers
LTV	Lifetime Value	ARPU × Average Customer Lifespan

Cost Metrics

Metric	Description	Source
Total COGS	Cost of Goods Sold	AWS + External AI + Platform Fees
Gross Profit	Revenue - COGS	Calculated
Gross Margin	(Gross Profit / Revenue) × 100	Percentage
CAC	Customer Acquisition Cost	Marketing + Sales / New Customers

Customer Metrics

Metric	Description
Total Customers	Active paying tenants
New Customers	Customers acquired in period
Churned Customers	Customers lost in period
Churn Rate	(Churned / Total) × 100
Net Revenue Retention	(Starting MRR + Expansion - Churn) / Starting MRR

Unit Economics

Metric	Healthy Range	Description
LTV:CAC Ratio	$> 3{:}1$	Lifetime value vs acquisition cost
Payback Period	< 12 months	Time to recover CAC
Gross Margin	$> 50\%$	COGS efficiency

Dashboard Tabs

Overview Tab

- Revenue, Cost & Profit trend chart (daily)
- Revenue by Source pie chart
- Revenue by Tier bar chart
- Cost breakdown with progress bars
- Top 5 tenants by revenue table

Revenue Tab

- MRR movement chart (New, Expansion, Churned)
- Revenue by Product breakdown
- ARPU and LTV metrics
- Revenue growth trends

Costs Tab

- Cost distribution pie chart
- Cost breakdown table by category
- Gross margin trends
- CAC metrics

Customers Tab

- Customer growth trend chart
- New vs Churned visualization
- Usage metrics (requests, active users)
- Churn rate tracking

Models Tab

- Model performance table (requests, revenue, cost, profit, margin)
- Model revenue vs cost horizontal bar chart
- Self-hosted vs External identification

Charts & Visualizations

Chart Types Used

Chart	Purpose
Area Chart	Revenue and cost trends over time
Composed Chart	Multi-metric comparisons
Bar Chart	Revenue by tier, model comparisons
Pie Chart	Revenue/cost distribution
Line Chart	Usage trends

Color Palette

```
const CHART_COLORS = {
  primary: '#3b82f6',     // Blue - Revenue
  secondary: '#8b5cf6',   // Purple - Secondary metrics
  success: '#22c55e',     // Green - Profit, positive
  warning: '#f59e0b',     // Amber - Warnings
  danger: '#ef4444',      // Red - Costs, negative
  info: '#06b6d4',        // Cyan - Info
};
```

Export Functionality

Available Formats

Format	Extension	Use Case
Excel (CSV)	`.csv`	Spreadsheet analysis, Excel/Google Sheets
JSON	`.json`	Custom integrations, data processing

Export Contents

The export includes all metrics organized in sections:

1. **Report Header**: Period, generation timestamp
2. **Revenue Summary**: MRR, ARR, Total Revenue, Gross Profit, Margin
3. **Cost Breakdown**: By category with amounts and percentages
4. **Customer Metrics**: Totals, new, churned, churn rate
5. **Unit Economics**: ARPU, LTV, CAC, LTV:CAC ratio
6. **Revenue by Source**: Subscriptions, Credits, AI Markup, etc.
7. **Revenue by Tier**: Enterprise, Business, Pro, Starter
8. **Top Tenants**: Name, revenue, users, requests, tier
9. **Model Performance**: Per-model revenue, cost, profit, margin
10. **Daily Revenue Trend**: Date, revenue, cost, profit

Export Example (CSV)

```
RADIANT SaaS Metrics Report
Period: 30d
Generated: 2024-12-28T22:57:00Z

=== REVENUE SUMMARY ===
Metric,Value,Growth
MRR,$89500.00,12.5%
ARR,$1074000.00,15.2%
Total Revenue,$89500.00,18.3%
Gross Profit,$47500.00,
Gross Margin,53.1%,

=== COST BREAKDOWN ===
Category,Amount,Percentage
External AI Providers,$18500.00,44.0%
AWS Compute,$12800.00,30.5%
...

=== MODEL PERFORMANCE ===
Model,Requests,Revenue,Cost,Profit,Margin
```

```
GPT-4o,425000,$12750.00,$8500.00,$4250.00,33.3%
Claude 3.5 Sonnet,312000,$9360.00,$6240.00,$3120.00,33.3%
```

Period Selection

Period	Description	Data Points
7d	Last 7 days	Daily granularity
30d	Last 30 days	Daily granularity
90d	Last 90 days	Daily granularity
12m	Last 12 months	Monthly granularity

API Integration

Data Sources

The SaaS Metrics dashboard aggregates data from:

1. **Revenue Analytics** (`/api/admin/revenue/dashboard`)
   - Revenue entries
   - Daily aggregates
   - Revenue by source
2. **Cost Analytics** (`/api/admin/cost/summary`)
   - Cost entries
   - AWS costs
   - External AI costs
3. **Billing System** (`/api/admin/billing`)
   - Subscriptions
   - Credit transactions
   - Tier information
4. **Analytics** (`/api/admin/analytics`)
   - Usage metrics
   - Model performance
   - Request counts

API Endpoint

`GET /api/admin/saas-metrics?period={7d|30d|90d|12m}`

Response:

```
{
  "mrr": 89500,
  "mrrGrowth": 12.5,
```

```
  "arr": 1074000,
  "totalRevenue": 89500,
  "totalCost": 42000,
  "grossProfit": 47500,
  "grossMargin": 53.1,
  "totalCustomers": 342,
  "churnRate": 2.3,
  "arpu": 261.70,
  "ltv": 3140.40,
  "cac": 450,
  "ltvCacRatio": 6.98,
  "revenueBySource": [...],
  "costByCategory": [...],
  "revenueTrend": [...],
  "topTenants": [...],
  "modelMetrics": [...]
}
```

Permissions

The SaaS Metrics Dashboard requires: - **Admin** role or higher - Access to billing data - Access to usage analytics

---

Related Documentation

- Revenue Analytics
- Cost Analytics
- Billing & Credits
- Admin Guide - Revenue Analytics

SPECIALTY-RANKING

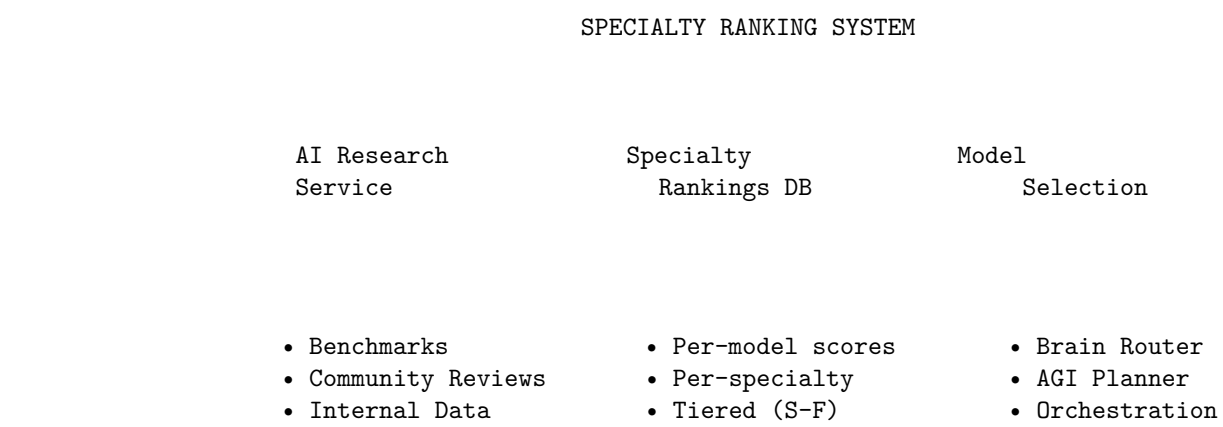
RADIANT Specialty Ranking System

Version: 4.18.0 Last Updated: 2024-12-28

Overview

The Specialty Ranking System is RADIANT's **AI-powered proficiency ranking** for models and orchestration modes. It provides domain-specific expertise scores that drive intelligent model selection.

Architecture



---

20 Specialty Categories

Models are ranked across 20 specialty categories representing domain-specific expertise:

Domain Expertise

Category	Icon	Description	Example Tasks
`medical`		Medical & Healthcare	Diagnosis, treatment, clinical guidelines
`legal`		Legal & Compliance	Contract review, legal research, compliance
`finance`		Finance & Trading	Financial analysis, trading strategies
`science`		Scientific	Research methodology, scientific writing
`security`		Cybersecurity	Vulnerability analysis, security audits
`architecture`		System Architecture	System design, scalability planning

Task Capabilities

Category	Icon	Description	Example Tasks
`reasoning`		Reasoning & Logic	Complex deduction, logical analysis
`coding`		Code Generation	Programming, debugging, refactoring
`math`		Mathematics	Calculations, proofs, statistics
`creative`		Creative Writing	Stories, poetry, marketing copy
`analysis`		Data Analysis	Data interpretation, patterns
`research`		Research & Synthesis	Literature review, synthesis
`debugging`		Debugging & QA	Bug finding, test generation
`conversation`		Conversational	Natural dialogue, engagement

Modalities

Category	Icon	Description	Example Tasks
`vision`		Vision & Images	Image analysis, OCR, diagrams
`audio`		Audio & Speech	Transcription, voice analysis

Performance Attributes

Category	Icon	Description	Example Tasks
`speed`		Low Latency	Real-time responses
`accuracy`		High Accuracy	Fact-critical tasks
`safety`		Safety & Alignment	Sensitive content handling
`instruction`		Instruction Following	Complex multi-step tasks

Tier System

Each model receives a tier rating (S-F) for each specialty:

Tier	Score Range	Description	Use Case
S	95-100	Elite - Best-in-class	Primary selection for this specialty
A	85-94	Excellent - Highly recommended	Strong choice, reliable
B	75-84	Good - Solid performance	Acceptable, cost-effective
C	65-74	Average - Acceptable	Use if better unavailable
D	50-64	Below Average - Use with caution	Fallback only
F	0-49	Poor - Not recommended	Do not use

Specialty Ranking Data Structure

```typescript
interface SpecialtyRanking {
  rankingId: string;
  modelId: string;                    // e.g., 'anthropic/claude-3-5-sonnet'
  provider: string;                   // e.g., 'anthropic'
  specialty: SpecialtyCategory;       // e.g., 'medical', 'coding'

  // Scores (0-100)
  proficiencyScore: number;           // Overall weighted score
  benchmarkScore: number;             // From published benchmarks
  communityScore: number;             // From community reviews
  internalScore: number;              // From internal usage data

  // Rankings
  rank: number;                       // Global rank for this specialty
  percentile: number;                 // e.g., 95 = top 5%
  tier: 'S' | 'A' | 'B' | 'C' | 'D' | 'F';

  // Metadata
  confidence: number;                 // 0-1 confidence in assessment
  dataPoints: number;                 // Number of data points used
  lastResearched: string;             // ISO timestamp
  researchSources: string[];          // Sources used
  trend: 'improving' | 'stable' | 'declining';

  // Admin
  adminOverride?: number;             // Locked admin score
```

```
  isLocked: boolean;                    // Whether ranking is locked
  updatedAt: string;
}
```

Mode Rankings

In addition to specialty rankings, models are ranked for each **orchestration mode**:

```
interface ModeRanking {
  rankingId: string;
  mode: OrchestrationMode;              // e.g., 'extended_thinking', 'coding'
  modelId: string;
  provider: string;
  score: number;
  tier: 'S' | 'A' | 'B' | 'C' | 'D' | 'F';
  strengths: string[];                  // What this model excels at
  weaknesses: string[];                 // Where it falls short
  recommendedFor: string[];             // Task types recommended for
  notRecommendedFor: string[];          // Task types to avoid
  confidence: number;
  isLocked: boolean;
  adminOverride?: number;
  updatedAt: string;
}
```

Orchestration Modes

Mode	Icon	Description
`thinking`		Standard reasoning with step-by-step analysis
`extended_thinking`		Deep multi-step reasoning for complex problems
`research`		Information gathering and synthesis
`creative`		Divergent thinking and idea generation
`analytical`		Data analysis and pattern recognition
`coding`		Code generation and debugging
`conversational`		Natural dialogue and engagement
`fast`		Quick responses with minimal latency
`precise`		High accuracy with verification
`balanced`		Optimal cost/quality/speed tradeoff

Model Specialty Profiles

Claude 3.5 Sonnet

```
Specialty Scores (0-100):
  reasoning:     94 (S)
  coding:        95 (S)
  math:          88 (A)
  creative:      92 (A)
  analysis:      91 (A)
  research:      90 (A)
  medical:       92 (A)
  legal:         89 (A)
  finance:       88 (A)
  security:      91 (A)
  vision:        93 (A)
  safety:        95 (S)
  speed:         75 (B)

Best For: General-purpose, coding, creative, research, analysis
Mode Recommendations: thinking, extended_thinking, creative, research
```

OpenAI o1

```
Specialty Scores (0-100):
  reasoning:     98 (S)
  coding:        90 (A)
  math:          96 (S)
  creative:      75 (B)
  analysis:      94 (S)
  research:      88 (A)
  medical:       85 (B)
  legal:         88 (A)
  finance:       91 (A)
  security:      89 (A)
  safety:        92 (A)
  speed:         60 (D)

Best For: Complex reasoning, mathematics, analysis, multi-step problems
Mode Recommendations: extended_thinking, analytical, precise
```

DeepSeek Coder

```
Specialty Scores (0-100):
  reasoning:     85 (B)
  coding:        96 (S)
  math:          92 (A)
```

```
  creative:      65 (C)
  analysis:      82 (B)
  debugging:     94 (S)
  architecture: 88 (A)
  security:      85 (B)
  speed:         90 (A)

Best For: Code generation, debugging, system design
Mode Recommendations: coding, fast
```

GPT-4o

```
Specialty Scores (0-100):
  reasoning:     90 (A)
  coding:        88 (A)
  math:          85 (B)
  creative:      88 (A)
  analysis:      86 (A)
  research:      87 (A)
  vision:        95 (S)
  audio:         92 (A)
  conversation: 91 (A)
  speed:         88 (A)

Best For: Multimodal tasks, vision, audio, conversation
Mode Recommendations: conversational, fast, balanced
```

Gemini 2.0 Flash

```
Specialty Scores (0-100):
  reasoning:     82 (B)
  coding:        80 (B)
  math:          78 (B)
  analysis:      80 (B)
  research:      82 (B)
  vision:        85 (B)
  speed:         98 (S)
  conversation: 85 (B)

Best For: Fast responses, real-time applications
Mode Recommendations: fast, conversational
```

AI-Powered Research

The specialty rankings are maintained through **automated AI research**:

Research Process

```
SPECIALTY RANKING RESEARCH FLOW



STEP 1: Gather Data Sources
• Published benchmarks (MMLU, HumanEval, MATH, GPQA, etc.)
• Community reviews (Reddit, Twitter, Discord)
• Academic papers and evaluations
• Internal usage data and quality scores



STEP 2: AI Analysis
• Claude 3.5 Sonnet analyzes all sources
• Generates per-specialty proficiency scores
• Assigns tier ratings (S/A/B/C/D/F)
• Calculates confidence levels



STEP 3: Score Calculation

proficiencyScore = (benchmarkWeight × benchmarkScore) +
                   (communityWeight × communityScore) +
                   (internalWeight × internalScore)

Default Weights: benchmark=0.5, community=0.3, internal=0.2



STEP 4: Update Rankings
• Update specialty_rankings table
• Recalculate global ranks per specialty
• Calculate percentiles
• Record research log
```

Research API

```
// Research a specific model across all specialties
const result = await specialtyRankingService.researchModelProficiency(
  'anthropic/claude-3-5-sonnet'
);
// Returns: { modelsResearched: 1, specialtiesUpdated: 20, rankingsChanged: 20 }

// Research all models for a specific specialty
const result = await specialtyRankingService.researchSpecialtyRankings('medical');
// Returns: { modelsResearched: 50, specialtiesUpdated: 1, rankingsChanged: 45 }

// Get leaderboard for a specialty
const leaderboard = await specialtyRankingService.getSpecialtyLeaderboard('coding', 10);
// Returns: { specialty: 'coding', rankings: [{ rank: 1, modelId: '...', score: 96, tier: '
```

```
// Get best model for a specialty
const best = await specialtyRankingService.getBestModelForSpecialty('medical', { minScore: 8
// Returns: { modelId: 'anthropic/claude-3-5-sonnet', score: 92, tier: 'A' }
```

Research Schedule

```
interface ResearchSchedule {
  scheduleId: string;
  name: string;
  frequency: 'hourly' | 'daily' | 'weekly' | 'monthly' | 'manual';
  cronExpression?: string;
  enabled: boolean;
  lastRun?: string;
  nextRun?: string;
  targetScope: 'all' | 'specialty' | 'mode' | 'model';
  targetFilter?: string;
}

// Example schedules:
// - Daily research for new models
// - Weekly refresh of all specialty rankings
// - Monthly deep research with expanded sources
```

Admin Controls

Admin Dashboard

Path: Admin Dashboard → Orchestration → Specialty Rankings

Features: - **Leaderboards**: View top models per specialty - **Model Profiles**:

See all specialty scores for a model - **Override Scores**: Lock a model's specialty score - **Trigger Research**: Manually refresh rankings - **Configure Weights**: Adjust scoring weights

Admin API

```
// Override a ranking (locks it from research updates)
await specialtyRankingService.adminOverrideRanking(
  'anthropic/claude-3-5-sonnet',
  'medical',
  95,  // New score
  'Internal evaluation showed higher medical accuracy'
);

// Unlock a ranking (allows research to update it again)
await specialtyRankingService.unlockRanking('anthropic/claude-3-5-sonnet', 'medical');

// Get model rankings
const rankings = await specialtyRankingService.getModelRankings('anthropic/claude-3-5-sonnet
```

Integration with Orchestration

Brain Router Integration

The Brain Router uses specialty rankings for model selection:

```
// In brain-router.ts
const bestMedicalModel = await specialtyRankingService.getBestModelForSpecialty('medical', {
  minScore: 85,
  excludeModels: disabledModels
});

// Factor specialty score into routing decision
const domainMatchScore = await getSpecialtyScore(modelId, detectedSpecialty);
const finalScore = costScore * 0.3 + latencyScore * 0.2 + qualityScore * 0.3 + domainMatchSc
```

AGI Brain Planner Integration

The AGI Brain Planner uses specialty rankings to select models:

```
// In agi-brain-planner.service.ts
const { primary, fallbacks } = await this.selectModels(
  tenantId,
  promptAnalysis,
  domainResult,      // Contains detected domain/subspecialty
  orchestrationMode
```

```
);

// Models are selected based on:
// 1. Domain proficiency match (from domain taxonomy)
// 2. Specialty rankings (from specialty ranking service)
// 3. Mode rankings (how well model performs in the chosen mode)
```

Combined Scoring Example

```
Prompt: "Review this contract for liability issues"

Domain Detection:
  Field: Law → Domain: Contract Law → Subspecialty: Commercial Contracts
  Confidence: 0.89

Required Specialties: legal, accuracy, reasoning

Model Scoring:

 Model                  Legal    Accuracy   Reasoning  Combined

 Claude 3.5 Sonnet    89 (A)   91 (A)     94 (S)      91.3
 GPT-4o               85 (B)   88 (A)     90 (A)      87.7
 OpenAI o1            88 (A)   92 (A)     98 (S)      92.7


Selected: OpenAI o1 (highest combined score for legal + reasoning)
```

Database Schema

```sql
-- Specialty rankings table
CREATE TABLE specialty_rankings (
  ranking_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  model_id TEXT NOT NULL,
  provider TEXT NOT NULL,
  specialty TEXT NOT NULL,
  proficiency_score NUMERIC(5,2) NOT NULL,
  benchmark_score NUMERIC(5,2),
  community_score NUMERIC(5,2),
  internal_score NUMERIC(5,2),
  rank INTEGER,
  percentile NUMERIC(5,2),
  tier TEXT NOT NULL CHECK (tier IN ('S', 'A', 'B', 'C', 'D', 'F')),
  confidence NUMERIC(3,2) DEFAULT 0.80,
  data_points INTEGER DEFAULT 0,
```

```
  last_researched TIMESTAMPTZ,
  research_sources TEXT[],
  trend TEXT DEFAULT 'stable' CHECK (trend IN ('improving', 'stable', 'declining')),
  admin_override NUMERIC(5,2),
  admin_notes TEXT,
  is_locked BOOLEAN DEFAULT false,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(model_id, specialty)
);

-- Mode rankings table
CREATE TABLE mode_rankings (
  ranking_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  mode TEXT NOT NULL,
  model_id TEXT NOT NULL,
  provider TEXT NOT NULL,
  score NUMERIC(5,2) NOT NULL,
  tier TEXT NOT NULL CHECK (tier IN ('S', 'A', 'B', 'C', 'D', 'F')),
  strengths TEXT[],
  weaknesses TEXT[],
  recommended_for TEXT[],
  not_recommended_for TEXT[],
  confidence NUMERIC(3,2) DEFAULT 0.80,
  admin_override NUMERIC(5,2),
  is_locked BOOLEAN DEFAULT false,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(mode, model_id)
);

-- Research logs
CREATE TABLE specialty_research_logs (
  log_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  research_type TEXT NOT NULL, -- 'model', 'specialty', 'full'
  target_id TEXT,
  models_researched INTEGER,
  specialties_updated INTEGER,
  rankings_changed INTEGER,
  duration_ms INTEGER,
  ai_confidence NUMERIC(3,2),
  sources_used TEXT[],
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

Related Documentation

- Orchestration Methods - Complete orchestration system documentation
- Domain Taxonomy - Domain detection and proficiency system
- AGI Brain Planner - Real-time planning system
- Model Router - Intelligent model selection

API Reference

Endpoints

Method	Path	Description
GET	`/api/admin/specialty-rankings`	List all rankings
GET	`/api/admin/specialty-rankings/:modelId`	Get rankings for model
GET	`/api/admin/specialty-rankings/specialty/:specialty`	Get rankings by specialty
POST	`/api/admin/specialty-rankings/research/model/:modelId`	Research model
POST	`/api/admin/specialty-rankings/research/specialty/:specialty`	Research specialty
PATCH	`/api/admin/specialty-rankings/:modelId/:specialty`	Override ranking
DELETE	`/api/admin/specialty-rankings/:modelId/:specialty/lock`	Unlock ranking
GET	`/api/admin/mode-rankings`	List mode rankings
GET	`/api/admin/mode-rankings/leaderboard`	Get leaderboard

THINK-TANK-EASTER-EGGS

Think Tank Easter Eggs Guide

Overview

Think Tank includes hidden easter eggs that provide fun, alternative interaction modes for users. Easter eggs are **Think Tank only** features - they are not available in the Radiant Admin dashboard except for administrative configuration.

Enabling/Disabling Easter Eggs

User Settings

Users can enable or disable easter eggs in Think Tank Settings: - Navigate to **Settings** → **Delight Preferences** - Toggle **Enable Easter Eggs** on/off

Easter eggs are enabled by default for users with `expressive` or `playful` personality modes.

Admin Configuration

Administrators manage easter eggs via: - **Admin Dashboard → Think Tank → Delight → Easter Eggs** - Individual easter eggs can be enabled/disabled - Discovery statistics are tracked per easter egg

---

Available Easter Eggs

Keyboard Triggered

Easter Egg	Trigger	Description	Duration
Konami Code	`↑↑↓↓←→←→BA`	Classic gaming mode with retro arcade theme	60 seconds

How to Activate: Press the arrow keys and letters in sequence: Up, Up, Down, Down, Left, Right, Left, Right, B, A

Activation Message: Cheat codes activated. +30 lives.

Text Command Triggered

Type these commands in the Think Tank chat input to activate:

Easter Egg	Command	Effect	Duration
Chaos Mode	`/chaos`	Models debate openly, disagreements visible	Until deactivated
Socratic Mode	`/socratic`	AI responds with questions instead of answers	Until deactivated
Victorian Gentleman	`/victorian`	Formal, Victorian-era speech style	Until deactivated
Pirate Mode	`/pirate`	Arrr! Pirate-speak responses	Until deactivated
Haiku Mode	`/haiku`	All responses in haiku format (5-7-5)	Until deactivated

Easter Egg	Command	Effect	Duration
Matrix Mode	`/matrix`	Green code rain visual effect	30 seconds
Disco Mode	`/disco`	Disco lights and music	30 seconds
Dad Jokes Mode	`/dadjokes`	Every response includes a dad joke	Until deactivated
Emission Mode	`/emissions`	Fun sound effects for events	Until deactivated

---

Detailed Easter Egg Descriptions

/chaos - Chaos Mode

Purpose: Let the AI models argue openly about their answers.

Activation: Type `/chaos` in the chat input

Effect: - Shows disagreements between models - Displays which models favor which approaches - More "raw" multi-model output

Deactivation: Type `/chaos` again or `/normal`

Message: Chaos Mode engaged. May the best model win.

/socratic - Socratic Mode

Purpose: The AI asks probing questions instead of giving direct answers.

Activation: Type `/socratic` in the chat input

Effect: - Responses are primarily questions - Encourages user to think through problems - Great for learning and exploration

Deactivation: Type `/socratic` again or `/normal`

Message: Socratic Mode. I'll ask the questions now.

/victorian - Victorian Gentleman Mode

Purpose: Formal, eloquent responses in Victorian-era style.

Activation: Type `/victorian` in the chat input

Effect: - Highly formal language - Victorian idioms and expressions - Polite, elaborate responses

Deactivation: Type `/victorian` again or `/normal`

Message: Indeed, good sir/madam. How may I assist?

/pirate - Pirate Mode

Purpose: Responses delivered in pirate-speak.

Activation: Type `/pirate` in the chat input

Effect: - "Arrr" and nautical terminology - Pirate idioms and phrases - Fun for casual conversations

Deactivation: Type `/pirate` again or `/normal`

Message: Ahoy! Ready to sail the seven seas of knowledge!

/haiku - Haiku Mode

Purpose: All responses formatted as haikus.

Activation: Type `/haiku` in the chat input

Effect: - 5-7-5 syllable structure - Poetic, condensed responses - Great for creative exploration

Deactivation: Type `/haiku` again or `/normal`

Message: Five, seven, then five / Syllables mark the rhythm / Nature finds its voice

---

/matrix - Matrix Mode

Purpose: Visual transformation with matrix-style code rain.

Activation: Type `/matrix` in the chat input

Effect: - Green falling code visual effect - Matrix-themed interface - Automatically ends after 30 seconds

Duration: 30 seconds (auto-deactivates)

Message: You took the red pill. Let's see how deep this goes.

/disco - Disco Mode

Purpose: Party atmosphere with lights and music.

Activation: Type `/disco` in the chat input

Effect: - Disco ball visual effects - Optional background music (if sounds enabled) - Automatically ends after 30 seconds

Duration: 30 seconds (auto-deactivates)

Message: Let's groove while we think!

/dadjokes - Dad Jokes Mode

Purpose: Every response includes a groan-worthy dad joke.

Activation: Type `/dadjokes` in the chat input

Effect: - Responses include related dad jokes - Puns and wordplay throughout - Warning: may cause eye-rolling

Deactivation: Type `/dadjokes` again or `/normal`

Message: Warning: Side effects include groaning and eye-rolling.

/emissions - Emission Mode

Purpose: Fun sound effects for various Think Tank events.

Activation: Type `/emissions` in the chat input

Effect: - Playful sound effects for: - Synthesis complete - Model agreement - Confirmations - Uses the "emissions" sound theme

Deactivation: Type `/emissions` again or `/normal`

Message: Emissions enabled. This is going to be fun.

Deactivating Easter Eggs

Method 1: Toggle Off

Type the same command again to toggle off: - `/pirate` → enables → `/pirate` → disables

Method 2: Return to Normal

Type `/normal` to return to standard mode and deactivate all active easter eggs.

Method 3: Automatic Timeout

Some easter eggs (Matrix, Disco) automatically deactivate after their duration expires.

Method 4: Settings

Disable all easter eggs via Settings → Delight Preferences → Enable Easter Eggs → Off

---

Achievement Integration

Discovering easter eggs contributes to achievements:

Achievement	Requirement	Reward
Curious One	Find 1 easter egg	20 points
Easter Hunter	Find 5 easter eggs	50 points

First-time discoveries are tracked and contribute to the discovery count displayed in admin analytics.

Admin-Only Notes

Easter eggs are managed exclusively through the Radiant Admin Dashboard:

- **View all easter eggs**: Admin Dashboard → Think Tank → Delight → Easter Eggs
- **Enable/disable individual eggs**: Toggle the enabled status
- **View discovery statistics**: See how many users have found each egg
- **Create custom easter eggs**: Add new triggers and effects

Easter egg functionality is **not exposed** in the main Radiant admin interface beyond configuration. They are a Think Tank consumer feature only.

API Reference

Trigger Easter Egg

```
POST /api/thinktank/delight/easter-egg/trigger
{
  "triggerType": "text_input" | "key_sequence" | "time_based" | "random" | "usage_pattern",
  "triggerValue": "/pirate"
}
```

Response

```
{
  "easterEgg": {
    "id": "pirate",
    "name": "Pirate Mode",
    "effectType": "mode_change",
    "effectConfig": { "mode": "pirate", "responseStyle": "pirate" },
    "activationMessage": "  Ahoy! Ready to sail the seven seas of knowledge!",
    "effectDurationSeconds": 0
  },
  "isNewDiscovery": true,
  "achievementUnlocked": null
}
```

Deactivate Easter Egg

```
POST /api/thinktank/delight/easter-egg/deactivate
{
  "easterEggId": "pirate"
}
```

Database Schema

Easter eggs are stored in the `delight_easter_eggs` table:

Column	Type	Description
`id`	VARCHAR(50)	Unique identifier
`name`	VARCHAR(100)	Display name
`trigger_type`	VARCHAR(30)	How it's triggered
`trigger_value`	TEXT	The trigger pattern/command
`effect_type`	VARCHAR(30)	What type of effect
`effect_config`	JSONB	Effect configuration
`effect_duration_seconds`	INTEGER	0 = until toggled off
`activation_message`	TEXT	Shown on activation

Column	Type	Description
`deactivation_message`	TEXT	Shown on deactivation
`is_enabled`	BOOLEAN	Admin toggle
`discovery_count`	INTEGER	Total discoveries

COMPLIANCE

RADIANT Compliance Guide

Overview

This document outlines RADIANT's compliance posture for SOC 2, HIPAA, and GDPR requirements.

Compliance Matrix

Framework	Tier Required	Status
SOC 2 Type II	All tiers	Controls implemented
HIPAA	Tier 3+ (GROWTH)	BAA available
GDPR	All tiers (EU data)	DPA available
PCI DSS	N/A	Not applicable (no card data)

SOC 2 Controls

Trust Service Criteria

Security (Common Criteria)

Control	Implementation
CC1.1 - Board oversight	Documented security policies
CC2.1 - Communication	Security awareness training
CC3.1 - Risk assessment	Annual risk assessments
CC4.1 - Monitoring	CloudWatch, GuardDuty
CC5.1 - Logical access	IAM, Cognito, RLS
CC6.1 - System operations	Runbooks, on-call
CC7.1 - Change management	CI/CD, PR reviews
CC8.1 - Risk mitigation	WAF, rate limiting
CC9.1 - Entity risk	Vendor assessments

Availability

Control	Implementation
A1.1 - Capacity planning	Auto-scaling, monitoring
A1.2 - Environmental protection	Multi-AZ, DR procedures
A1.3 - Recovery	Backups, PITR, runbooks

Confidentiality

Control	Implementation
C1.1 - Data classification	PII tagging, encryption
C1.2 - Data disposal	Lifecycle policies

Evidence Collection

```
// Automated evidence collection
const auditLogs = {
  // All admin actions logged
  source: 'audit_logs table',
  retention: '7 years',

  // Access logs
  accessLogs: 'CloudWatch Logs',

  // Configuration changes
  configChanges: 'AWS Config',

  // Security events
  securityEvents: 'GuardDuty findings',
};
```

Annual Audit Checklist

- ☐ Access review completed
- ☐ Penetration test completed
- ☐ Vulnerability scan completed
- ☐ Security training completed
- ☐ Incident response test completed
- ☐ DR test completed
- ☐ Vendor assessments updated
- ☐ Policies reviewed and updated

HIPAA Compliance

Applicability

HIPAA compliance is available for Tier 3 (GROWTH) and above, which includes: - Encryption at rest (AES-256) - Encryption in transit (TLS 1.3) - Audit logging - Access controls - BAA with AWS

Technical Safeguards

Requirement	Implementation
Access Control (§164.312(a))	Cognito MFA, RLS, RBAC
Audit Controls (§164.312(b))	CloudTrail, audit_logs table
Integrity Controls (§164.312(c))	Checksums, versioning
Transmission Security (§164.312(e))	TLS 1.3, VPC endpoints

Administrative Safeguards

Requirement	Implementation
Security Officer	Designated in org
Workforce Training	Annual security training
Access Management	Quarterly access reviews
Incident Response	Documented procedures

Physical Safeguards

Handled by AWS: - Data center security - Device controls - Facility access

PHI Data Handling

```
-- PHI fields are encrypted at column level
CREATE TABLE patient_data (
  id UUID PRIMARY KEY,
  tenant_id UUID NOT NULL,
  -- PHI fields use additional encryption
  encrypted_data BYTEA NOT NULL,
  encryption_key_id VARCHAR(255) NOT NULL,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Enable RLS for tenant isolation
ALTER TABLE patient_data ENABLE ROW LEVEL SECURITY;
```

BAA Requirements

Before processing PHI: 1. Sign BAA with RADIANT 2. Enable HIPAA-eligible services only 3. Configure CloudTrail logging 4. Enable AWS Config 5. Review shared responsibility model

GDPR Compliance

Data Subject Rights

Right	Implementation
Right to Access	Data export API
Right to Rectification	Self-service + API
Right to Erasure	Deletion API + cascade
Right to Restrict	Processing flags
Right to Portability	JSON/CSV export
Right to Object	Consent management

Data Export (Right to Access)

```
// API endpoint for data export
// GET /api/v2/gdpr/export
async function exportUserData(userId: string): Promise<UserDataExport> {
  return {
    personalData: await getPersonalData(userId),
    activityLogs: await getActivityLogs(userId),
    preferences: await getPreferences(userId),
    exportedAt: new Date().toISOString(),
    format: 'JSON',
  };
}
```

Data Deletion (Right to Erasure)

```
// API endpoint for data deletion
// DELETE /api/v2/gdpr/delete
async function deleteUserData(userId: string): Promise<DeletionResult> {
  // Cascade delete all user data
  await deletePersonalData(userId);
  await deleteActivityLogs(userId);
  await deletePreferences(userId);
  await deleteApiKeys(userId);

  // Anonymize audit logs (retain for compliance)
  await anonymizeAuditLogs(userId);
```

```
  return {
    deletedAt: new Date().toISOString(),
    confirmation: generateDeletionCertificate(userId),
  };
}
```

Data Processing Agreement

DPA includes: - Nature and purpose of processing - Types of personal data - Categories of data subjects - Sub-processor list - Technical measures - Audit rights

Data Residency

Region	Data Location	Backup Location
EU	eu-west-1 (Ireland)	eu-central-1 (Frankfurt)
US	us-east-1 (Virginia)	us-west-2 (Oregon)
APAC	ap-northeast-1 (Tokyo)	ap-southeast-1 (Singapore)

```
// Enforce data residency
const dataResidency = {
  EU: ['eu-west-1', 'eu-central-1'],
  US: ['us-east-1', 'us-west-2'],
  APAC: ['ap-northeast-1', 'ap-southeast-1'],
};

// Route requests to appropriate region
function routeByResidency(tenantRegion: string): string {
  return dataResidency[tenantRegion][0];
}
```

Consent Management

```
-- Consent tracking table
CREATE TABLE consent_records (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID NOT NULL REFERENCES users(id),
  consent_type VARCHAR(50) NOT NULL,
  granted BOOLEAN NOT NULL,
  granted_at TIMESTAMPTZ,
  withdrawn_at TIMESTAMPTZ,
  ip_address INET,
  user_agent TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW()
);
```

```
-- Consent types: marketing, analytics, essential, third_party
```

Data Classification

Classification Levels

Level	Description	Examples	Controls
Public	No restrictions	Marketing content	None
Internal	Business use	Metrics, configs	Access control
Confidential	Sensitive business	API keys, billing	Encryption, audit
Restricted	Highly sensitive	PHI, PII, credentials	Full controls

PII Fields

```
// Fields classified as PII
const piiFields = [
  'email',
  'display_name',
  'phone_number',
  'ip_address',
  'user_agent',
  'billing_address',
  'payment_method',
];

// Automatic PII detection and tagging
function tagPiiFields(data: Record<string, unknown>): void {
  for (const field of piiFields) {
    if (data[field]) {
      // Tag for audit and retention policies
      data[`${field}_pii`] = true;
    }
  }
}
```

Encryption

At Rest

Data Type	Encryption	Key Management
Database	AES-256	AWS KMS
S3	AES-256	AWS KMS
Secrets	AES-256	Secrets Manager
Backups	AES-256	AWS KMS

In Transit

Connection	Protocol	Minimum Version
API	TLS	1.2 (1.3 preferred)
Database	TLS	1.2
Internal	TLS	1.2

Key Rotation

```
// Automatic key rotation
const kmsKey = new kms.Key(this, 'Key', {
  enableKeyRotation: true,  // Annual rotation
  rotationPeriod: cdk.Duration.days(365),
});
```

Audit Logging

What We Log

Event Type	Retention	Purpose
Authentication	2 years	Security
Authorization	2 years	Security
Data access	7 years	Compliance
Admin actions	7 years	Compliance
Configuration changes	7 years	Compliance
API requests	90 days	Operations

Log Format

```
{
  "timestamp": "2024-12-24T10:30:00Z",
  "event_type": "data_access",
  "actor": {
    "id": "user-123",
    "type": "admin",
    "ip": "192.168.1.100"
```

```
  },
  "resource": {
    "type": "model",
    "id": "model-456"
  },
  "action": "read",
  "outcome": "success",
  "metadata": {}
}
```

Log Protection

- Logs are immutable (write-once)
- Logs are encrypted at rest
- Access requires special IAM role
- Log deletion requires dual approval

Incident Response

Classification

Severity	Response Time	Examples
Critical	1 hour	Data breach, service down
High	4 hours	Attempted breach, partial outage
Medium	24 hours	Policy violation
Low	72 hours	Minor security event

Breach Notification

Jurisdiction	Requirement	Timeline
GDPR	DPA + affected users	72 hours
HIPAA	HHS + affected individuals	60 days
State laws	Varies by state	Varies

Vendor Management

Approved Sub-Processors

Vendor	Purpose	Location	DPA
AWS	Infrastructure	Global	Yes
OpenAI	AI provider	US	Yes
Anthropic	AI provider	US	Yes
Google Cloud	AI provider	Global	Yes

Vendor	Purpose	Location	DPA

Vendor Assessment

Annual assessment includes: - Security questionnaire - SOC 2 report review - Penetration test results - Insurance verification

Contact

Role	Contact
Data Protection Officer	dpo@radiant.example.com
Security Team	security@radiant.example.com
Compliance Team	compliance@radiant.example.com

DATA_RETENTION

RADIANT Data Retention Policy

Overview

This document defines data retention periods and deletion procedures for all data stored in the RADIANT platform.

Retention Schedule

User Data

Data Type	Active Retention	Archive	Total Retention	Deletion
Account info	Active + 30 days	N/A	Account lifetime + 30 days	Automatic
Usage history	2 years	5 years	7 years	Automatic
Chat history	90 days	1 year	1 year	Automatic
Uploaded files	Active	30 days post-delete	Active + 30 days	On request
API keys	Active	N/A	Revoked + 90 days	Automatic

System Data

Data Type	Retention	Purpose	Deletion
Audit logs	7 years	Compliance	Automatic
Access logs	2 years	Security	Automatic
Error logs	90 days	Debugging	Automatic
Metrics	15 months	CloudWatch default	Automatic
Backups	35 days	Recovery	Automatic

Billing Data

Data Type	Retention	Purpose	Legal Basis
Invoices	7 years	Tax compliance	Legal requirement
Transactions	7 years	Financial audit	Legal requirement
Payment methods	Active	Processing	Contract
Receipts	7 years	Tax compliance	Legal requirement

Implementation

Database Retention

```
-- Automatic data cleanup job (runs daily)
CREATE OR REPLACE FUNCTION cleanup_expired_data()
RETURNS void AS $$
BEGIN
  -- Delete expired chat messages (90 days)
  DELETE FROM chat_messages
  WHERE created_at < NOW() - INTERVAL '90 days'
  AND archived = false;

  -- Archive chat messages older than 90 days
  UPDATE chat_messages
  SET archived = true, archived_at = NOW()
  WHERE created_at < NOW() - INTERVAL '90 days'
  AND archived = false;

  -- Delete archived messages older than 1 year
  DELETE FROM chat_messages
  WHERE archived = true
  AND archived_at < NOW() - INTERVAL '1 year';

  -- Delete revoked API keys (90 days after revocation)
  DELETE FROM api_keys
```

```
  WHERE revoked_at < NOW() - INTERVAL '90 days';

  -- Delete expired sessions
  DELETE FROM user_sessions
  WHERE expires_at < NOW();

  -- Log cleanup
  INSERT INTO system_jobs (job_name, completed_at, records_affected)
  VALUES ('cleanup_expired_data', NOW(),
    (SELECT count(*) FROM pg_stat_user_tables WHERE relname IN
      ('chat_messages', 'api_keys', 'user_sessions')));
END;
$$ LANGUAGE plpgsql;

-- Schedule daily at 3 AM UTC
SELECT cron.schedule('data-cleanup', '0 3 * * *', 'SELECT cleanup_expired_data()');
```

S3 Lifecycle Policies

```
const bucket = new s3.Bucket(this, 'Storage', {
  lifecycleRules: [
    // User uploads - delete 30 days after object deletion marker
    {
      id: 'delete-old-versions',
      noncurrentVersionExpiration: cdk.Duration.days(30),
    },

    // Temp files - delete after 7 days
    {
      id: 'cleanup-temp',
      prefix: 'temp/',
      expiration: cdk.Duration.days(7),
    },

    // Logs - transition to Glacier after 90 days, delete after 2 years
    {
      id: 'archive-logs',
      prefix: 'logs/',
      transitions: [
        {
          storageClass: s3.StorageClass.GLACIER,
          transitionAfter: cdk.Duration.days(90),
        },
      ],
      expiration: cdk.Duration.days(730), // 2 years
    },
```

```
    // Backups - delete after 35 days
    {
      id: 'cleanup-backups',
      prefix: 'backups/',
      expiration: cdk.Duration.days(35),
    },
  ],
});
```

CloudWatch Log Retention

```
// Set retention for all log groups
const logRetention: Record<string, logs.RetentionDays> = {
  // Application logs
  '/aws/lambda/radiant-*': logs.RetentionDays.THREE_MONTHS,

  // API Gateway logs
  '/aws/apigateway/radiant-*': logs.RetentionDays.THREE_MONTHS,

  // Database logs (longer for compliance)
  '/aws/rds/cluster/radiant-*': logs.RetentionDays.TWO_YEARS,

  // Audit logs (longest retention)
  '/radiant/audit/*': logs.RetentionDays.TEN_YEARS,
};
```

Data Deletion

User-Initiated Deletion

Account Deletion Flow

```
async function deleteUserAccount(userId: string): Promise<void> {
  // 1. Verify identity (MFA required)
  await verifyIdentity(userId);

  // 2. Cancel active subscriptions
  await cancelSubscriptions(userId);

  // 3. Export data (optional, user-requested)
  const exportUrl = await exportUserData(userId);

  // 4. Mark account for deletion (30-day grace period)
  await markForDeletion(userId, {
    scheduledAt: addDays(new Date(), 30),
    reason: 'user_requested',
```

```
  });

  // 5. Send confirmation email
  await sendDeletionConfirmation(userId, exportUrl);
}

// Actual deletion after grace period
async function executeAccountDeletion(userId: string): Promise<void> {
  // Delete in order (respect foreign keys)
  await deleteApiKeys(userId);
  await deleteChatHistory(userId);
  await deleteFiles(userId);
  await deletePreferences(userId);
  await deleteBillingHistory(userId); // Anonymize, don't delete
  await deleteAccount(userId);

  // Anonymize audit logs
  await anonymizeAuditLogs(userId);

  // Log deletion for compliance
  await logAccountDeletion(userId);
}
```

Data Categories Deleted

Category	Action	Timing
Profile	Delete	Immediate
Preferences	Delete	Immediate
Chat history	Delete	Immediate
Files	Delete	Immediate
API keys	Revoke + Delete	Immediate
Billing history	Anonymize	Immediate
Audit logs	Anonymize	Immediate
Backups	Excluded	Expires naturally

Administrative Deletion

```
// Bulk deletion for compliance (e.g., GDPR request)
async function adminBulkDelete(
  tenantId: string,
  options: {
    dataTypes: string[];
    olderThan: Date;
    reason: string;
    approvedBy: string[];
```

```
  }
): Promise<DeletionReport> {
  // Require dual admin approval
  if (options.approvedBy.length < 2) {
    throw new Error('Dual admin approval required');
  }

  // Log the deletion request
  await logAdminAction({
    action: 'bulk_delete',
    tenantId,
    options,
  });

  // Execute deletion
  const results = await Promise.all(
    options.dataTypes.map(type =>
      deleteDataByType(tenantId, type, options.olderThan)
    )
  );

  return {
    requestId: generateRequestId(),
    deletedAt: new Date(),
    recordsDeleted: results.reduce((a, b) => a + b, 0),
    dataTypes: options.dataTypes,
  };
}
```

Tenant Offboarding

```
async function offboardTenant(tenantId: string): Promise<void> {
  // 1. Export all data (required for compliance)
  const exportUrl = await exportTenantData(tenantId);

  // 2. Notify all users
  await notifyTenantUsers(tenantId, 'account_closing');

  // 3. Wait for grace period (30 days default)
  await scheduleTenantDeletion(tenantId, {
    gracePeriod: 30,
    exportUrl,
  });

  // 4. After grace period, delete all data
  // (Handled by scheduled job)
```

```
}
```

Legal Holds

Implementing a Legal Hold

```
-- Legal hold table
CREATE TABLE legal_holds (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID REFERENCES tenants(id),
  user_id UUID REFERENCES users(id),
  hold_type VARCHAR(50) NOT NULL, -- 'litigation', 'investigation', 'regulatory'
  description TEXT,
  started_at TIMESTAMPTZ DEFAULT NOW(),
  expires_at TIMESTAMPTZ,
  created_by UUID REFERENCES administrators(id),
  CONSTRAINT legal_holds_target CHECK (tenant_id IS NOT NULL OR user_id IS NOT NULL)
);

-- Prevent deletion of held data
CREATE OR REPLACE FUNCTION check_legal_hold()
RETURNS TRIGGER AS $$
BEGIN
  IF EXISTS (
    SELECT 1 FROM legal_holds
    WHERE (tenant_id = OLD.tenant_id OR user_id = OLD.user_id)
    AND (expires_at IS NULL OR expires_at > NOW())
  ) THEN
    RAISE EXCEPTION 'Cannot delete data under legal hold';
  END IF;
  RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Suspending Retention Policies

```
// Suspend automatic deletion during legal hold
async function applyLegalHold(params: {
  holdId: string;
  scope: 'tenant' | 'user';
  targetId: string;
}): Promise<void> {
  // Update retention flags
  await updateRetentionPolicy(params.targetId, {
    suspended: true,
    holdId: params.holdId,
  });
```

```
  // Exclude from cleanup jobs
  await excludeFromCleanup(params.targetId);

  // Notify compliance team
  await notifyCompliance('legal_hold_applied', params);
}
```

Audit Trail

Retention Actions Log

```
-- Log all retention-related actions
CREATE TABLE retention_actions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  action_type VARCHAR(50) NOT NULL, -- 'delete', 'archive', 'export', 'hold'
  target_type VARCHAR(50) NOT NULL, -- 'user', 'tenant', 'data_type'
  target_id VARCHAR(255) NOT NULL,
  records_affected INTEGER,
  performed_by UUID,
  reason TEXT,
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Index for compliance queries
CREATE INDEX idx_retention_actions_date ON retention_actions(created_at);
CREATE INDEX idx_retention_actions_target ON retention_actions(target_type, target_id);
```

Compliance Reporting

```
// Generate retention compliance report
async function generateRetentionReport(
  startDate: Date,
  endDate: Date
): Promise<RetentionReport> {
  return {
    period: { start: startDate, end: endDate },

    // Data deleted by type
    deletions: await getRetentionActions('delete', startDate, endDate),

    // Data archived
    archives: await getRetentionActions('archive', startDate, endDate),

    // Active legal holds
    legalHolds: await getActiveLegalHolds(),
```

```
    // Policy violations (data past retention not deleted)
    violations: await getRetentionViolations(),

    // User deletion requests
    userRequests: await getUserDeletionRequests(startDate, endDate),
  };
}
```

Verification

Monthly Retention Audit

- ☐ Verify cleanup jobs running successfully
- ☐ Check for retention policy violations
- ☐ Review legal holds status
- ☐ Verify S3 lifecycle policies active
- ☐ Confirm CloudWatch log retention settings
- ☐ Review user deletion requests processed
- ☐ Update retention schedule if needed

Compliance Queries

```
-- Find data past retention period
SELECT
  'chat_messages' as table_name,
  COUNT(*) as records,
  MIN(created_at) as oldest_record
FROM chat_messages
WHERE created_at < NOW() - INTERVAL '90 days'
AND archived = false

UNION ALL

SELECT
  'api_keys' as table_name,
  COUNT(*) as records,
  MIN(revoked_at) as oldest_record
FROM api_keys
WHERE revoked_at < NOW() - INTERVAL '90 days';
```

Contact

Role	Contact	Purpose
Data Protection Officer	dpo@radiant.example.com	GDPR requests
Legal	legal@radiant.example.com	Legal holds

Role	Contact	Purpose
Compliance	compliance@radiant.example.com	Audit questions

DISASTER_RECOVERY

RADIANT Disaster Recovery Guide

Overview

This document outlines disaster recovery (DR) procedures for the RADIANT platform, including backup strategies, recovery procedures, and business continuity plans.

Recovery Objectives

Metric	Target	Maximum
RTO (Recovery Time Objective)	1 hour	4 hours
RPO (Recovery Point Objective)	5 minutes	1 hour

Backup Strategy

Database Backups

Automated Backups (Aurora)

```
// CDK Configuration
const database = new rds.DatabaseCluster(this, 'Database', {
  backup: {
    retention: cdk.Duration.days(35),      // 35 days retention
    preferredWindow: '03:00-04:00',        // 3-4 AM UTC
  },
  deletionProtection: true,
  storageEncrypted: true,
});
```

Point-in-Time Recovery Aurora supports point-in-time recovery (PITR) to any second within the retention period.

```
# Restore to specific point in time
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-production \
  --db-cluster-identifier radiant-production-restored \
```

```
  --restore-to-time "2024-12-24T10:30:00Z" \
  --vpc-security-group-ids sg-xxx \
  --db-subnet-group-name radiant-production
```

Manual Snapshots

```
# Create manual snapshot before major changes
aws rds create-db-cluster-snapshot \
  --db-cluster-identifier radiant-production \
  --db-cluster-snapshot-identifier radiant-production-pre-migration-$(date +%Y%m%d)
```

S3 Backups

Versioning

```
// All S3 buckets have versioning enabled
const bucket = new s3.Bucket(this, 'Storage', {
  versioned: true,
  lifecycleRules: [
    {
      noncurrentVersionExpiration: cdk.Duration.days(90),
    },
  ],
});
```

Cross-Region Replication

```
// Production buckets replicate to DR region
const replicationRule = {
  destination: {
    bucket: drBucket.bucketArn,
    storageClass: s3.StorageClass.STANDARD_IA,
  },
  status: 'Enabled',
};
```

Secrets Backup

```
# Export secrets for DR (store securely!)
aws secretsmanager get-secret-value \
  --secret-id radiant-production-db \
  --query SecretString \
  --output text > /secure/path/db-credentials.json
```

Failure Scenarios

Scenario 1: Single AZ Failure

Impact: Partial service degradation **Recovery:** Automatic (Multi-AZ)

Aurora automatically fails over to a read replica in another AZ.

```
# Monitor failover
aws rds describe-events \
  --source-type db-cluster \
  --source-identifier radiant-production \
  --duration 60
```

Scenario 2: Database Corruption

Impact: Data integrity issues **Recovery:** Point-in-time restore

1. Identify corruption time
2. Restore to point before corruption
3. Validate data integrity
4. Switch traffic to restored database

```
# Step 1: Identify issue time from logs
aws logs filter-log-events \
  --log-group-name /aws/rds/cluster/radiant-production/error \
  --start-time $(date -d '24 hours ago' +%s000)

# Step 2: Restore
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-production \
  --db-cluster-identifier radiant-dr-$(date +%Y%m%d%H%M) \
  --restore-to-time "2024-12-24T09:00:00Z"

# Step 3: Update Lambda environment to use new cluster
aws lambda update-function-configuration \
  --function-name radiant-production-router \
  --environment "Variables={DB_CLUSTER_ARN=arn:aws:rds:...}"
```

Scenario 3: Region Failure

Impact: Complete service outage **Recovery:** Failover to DR region

1. Activate DR region infrastructure
2. Promote Aurora Global Database secondary
3. Update Route 53 to point to DR region
4. Verify service health

```
# Step 1: Promote DR database
aws rds failover-global-cluster \
```

```
  --global-cluster-identifier radiant-global \
  --target-db-cluster-identifier radiant-dr-cluster

# Step 2: Update DNS
aws route53 change-resource-record-sets \
  --hosted-zone-id Z123456 \
  --change-batch file://dr-dns-failover.json
```

Scenario 4: Accidental Deletion

Impact: Data loss **Recovery:** Restore from backup

```
# Restore deleted S3 objects
aws s3api list-object-versions \
  --bucket radiant-storage-production \
  --prefix "deleted/path/" \
  --query 'DeleteMarkers[?IsLatest==`true`]'

# Restore specific version
aws s3api delete-object \
  --bucket radiant-storage-production \
  --key "path/to/file" \
  --version-id "delete-marker-version-id"
```

Scenario 5: Security Breach

Impact: Potential data exposure **Recovery:** Isolation and investigation

1. Isolate affected systems
2. Rotate all credentials
3. Investigate scope
4. Restore from known-good backup
5. Notify affected parties

```
# Step 1: Disable API access
aws apigateway update-stage \
  --rest-api-id abc123 \
  --stage-name v2 \
  --patch-operations op=replace,path=/throttling/rateLimit,value=0

# Step 2: Rotate database credentials
aws secretsmanager rotate-secret \
  --secret-id radiant-production-db

# Step 3: Invalidate all sessions
aws cognito-idp admin-user-global-sign-out \
  --user-pool-id us-east-1_xxx \
  --username "*"
```

Recovery Procedures

Database Recovery Runbook

```
#!/bin/bash
# database-recovery.sh

set -e

CLUSTER_ID="radiant-production"
RESTORE_TIME="${1:-$(date -d '1 hour ago' -Iseconds)}"
NEW_CLUSTER_ID="radiant-dr-$(date +%Y%m%d%H%M)"

echo "  Starting database recovery..."
echo "   Source: $CLUSTER_ID"
echo "   Restore time: $RESTORE_TIME"
echo "   New cluster: $NEW_CLUSTER_ID"

# Create restored cluster
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier "$CLUSTER_ID" \
  --db-cluster-identifier "$NEW_CLUSTER_ID" \
  --restore-to-time "$RESTORE_TIME" \
  --db-subnet-group-name radiant-production \
  --vpc-security-group-ids sg-xxx

echo "  Waiting for cluster to be available..."
aws rds wait db-cluster-available \
  --db-cluster-identifier "$NEW_CLUSTER_ID"

# Create instance
aws rds create-db-instance \
  --db-instance-identifier "${NEW_CLUSTER_ID}-instance-1" \
  --db-cluster-identifier "$NEW_CLUSTER_ID" \
  --db-instance-class db.r6g.large \
  --engine aurora-postgresql

echo "  Waiting for instance to be available..."
aws rds wait db-instance-available \
  --db-instance-identifier "${NEW_CLUSTER_ID}-instance-1"

echo "  Database restored successfully!"
echo "   Endpoint: $(aws rds describe-db-clusters \
  --db-cluster-identifier "$NEW_CLUSTER_ID" \
  --query 'DBClusters[0].Endpoint' --output text)"
```

Full Service Recovery Runbook

```
#!/bin/bash
# full-recovery.sh

set -e

echo "  RADIANT Full Service Recovery"
echo "================================="

# Step 1: Database
echo "Step 1: Recovering database..."
./scripts/dr/database-recovery.sh

# Step 2: Update Lambda configurations
echo "Step 2: Updating Lambda configurations..."
for fn in router admin billing localization configuration; do
  aws lambda update-function-configuration \
    --function-name "radiant-production-$fn" \
    --environment "Variables={DB_CLUSTER_ARN=$NEW_DB_ARN}"
done

# Step 3: Clear caches
echo "Step 3: Clearing caches..."
redis-cli -h radiant-cache.xxx.cache.amazonaws.com FLUSHALL

# Step 4: Verify health
echo "Step 4: Verifying service health..."
curl -f https://api.radiant.example.com/v2/health || exit 1

# Step 5: Run smoke tests
echo "Step 5: Running smoke tests..."
k6 run --env BASE_URL=https://api.radiant.example.com tests/load/k6-config.js

echo "  Recovery complete!"
```

Testing DR Procedures

Quarterly DR Drill

1. **Preparation**
   - Schedule maintenance window
   - Notify stakeholders
   - Prepare rollback plan
2. **Execution**
   - Simulate failure scenario
   - Execute recovery procedures

- Measure RTO/RPO

3. **Validation**
   - Verify data integrity
   - Run integration tests
   - Check all services
4. **Documentation**
   - Record actual RTO/RPO
   - Document issues encountered
   - Update procedures

DR Test Checklist

- ☐ Database point-in-time recovery tested
- ☐ S3 object recovery tested
- ☐ Secret rotation tested
- ☐ Lambda rollback tested
- ☐ DNS failover tested
- ☐ Communication plan executed
- ☐ Recovery time recorded
- ☐ Post-mortem completed

Communication Plan

Escalation Matrix

Severity	Response Time	Notify
SEV1	15 min	Eng Lead, CTO, Status Page
SEV2	30 min	Eng Lead, Status Page
SEV3	2 hours	On-call team

Status Page Updates

```
# Update status page (example with Statuspage.io)
curl -X POST https://api.statuspage.io/v1/pages/xxx/incidents \
  -H "Authorization: OAuth $STATUSPAGE_API_KEY" \
  -d '{
    "incident": {
      "name": "Service Degradation",
      "status": "investigating",
      "body": "We are investigating reports of API errors."
    }
  }'
```

Infrastructure as Code

All DR infrastructure is defined in CDK:

```typescript
// lib/stacks/dr-stack.ts
export class DRStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props: DRStackProps) {
    super(scope, id, props);

    // Global Database for cross-region replication
    const globalCluster = new rds.CfnGlobalCluster(this, 'GlobalCluster', {
      globalClusterIdentifier: 'radiant-global',
      sourceDbClusterIdentifier: props.primaryClusterArn,
    });

    // S3 Cross-Region Replication
    const drBucket = new s3.Bucket(this, 'DRBucket', {
      bucketName: `radiant-storage-dr-${props.drRegion}`,
    });
  }
}
```

Contacts

Role	Contact	Backup
DR Coordinator	dr@radiant.example.com	cto@radiant.example.com
Database Admin	dba@radiant.example.com	platform@radiant.example.com
Security	security@radiant.example.com	cto@radiant.example.com

---

COST_OPTIMIZATION

RADIANT Cost Optimization Guide

Overview

This guide provides strategies for optimizing AWS costs for the RADIANT platform while maintaining performance and reliability.

Current Architecture Costs

Estimated Monthly Costs by Tier

Tier	Infrastructure	Est. Monthly Cost
SEED (Dev)	Minimal	$50-150
STARTUP	Small production	$200-400
GROWTH	Self-hosted models	$1,000-2,500

Tier	Infrastructure	Est. Monthly Cost
SCALE	Multi-region	$4,000-8,000
ENTERPRISE	Global, full HA	$15,000-35,000

Cost Breakdown by Service

Service	% of Total	Optimization Potential
Aurora	30-40%	High
Lambda	15-25%	Medium
API Gateway	5-10%	Low
S3	5-10%	Medium
CloudFront	5-10%	Low
ElastiCache	10-15%	Medium
Other	10-15%	Varies

Optimization Strategies

1. Database Optimization

Aurora Serverless v2

```
// Use Serverless v2 for variable workloads
const cluster = new rds.DatabaseCluster(this, 'Database', {
  serverlessV2MinCapacity: 0.5,   // Scale to near-zero
  serverlessV2MaxCapacity: 16,    // Scale up when needed
});
```

Savings: 40-60% vs. provisioned instances for variable workloads

Reserved Instances (Steady Workloads)

```
# Purchase reserved capacity for predictable workloads
aws rds purchase-reserved-db-instances-offering \
  --reserved-db-instances-offering-id xxx \
  --db-instance-count 1
```

Savings: 30-60% for 1-3 year terms

Read Replicas Strategy

```
// Use read replicas only when needed
// Scale readers with traffic
readers: [
  rds.ClusterInstance.serverlessV2('reader', {
    scaleWithWriter: true,  // Auto-scale with primary
```

```
  }),
],
```

2. Lambda Optimization

Right-Size Memory

```
// Test different memory sizes to find optimal cost/performance
const memoryOptions = [256, 512, 1024, 2048];

// Use AWS Lambda Power Tuning tool
// https://github.com/alexcasalboni/aws-lambda-power-tuning
```

Function Type	Recommended Memory	Reason
Simple CRUD	256-512 MB	Light compute
API Router	512-1024 MB	Balanced
AI Processing	1024-2048 MB	Heavy compute

Provisioned Concurrency (Strategic)

```
// Only use for latency-critical functions
new lambda.Alias(this, 'LiveAlias', {
  aliasName: 'live',
  version: fn.currentVersion,
  provisionedConcurrentExecutions: 5,  // Keep 5 warm
});
```

Cost: ~$0.015/hour per provisioned instance **Use when:** P99 latency requirements $<$ 200ms

ARM64 (Graviton2)

```
// 20% cheaper, often faster
const fn = new lambda.Function(this, 'Function', {
  architecture: lambda.Architecture.ARM_64,
  runtime: lambda.Runtime.NODEJS_20_X,
});
```

Savings: 20% on compute costs

3. S3 Optimization

Intelligent Tiering

```
const bucket = new s3.Bucket(this, 'Storage', {
  intelligentTieringConfigurations: [{
    name: 'auto-tier',
    archiveAccessTierTime: cdk.Duration.days(90),
```

```
    deepArchiveAccessTierTime: cdk.Duration.days(180),
  }],
});
```

Savings: Up to 95% for infrequently accessed data

Lifecycle Rules

```
const bucket = new s3.Bucket(this, 'Storage', {
  lifecycleRules: [
    // Move old versions to cheaper storage
    {
      noncurrentVersionTransitions: [
        {
          storageClass: s3.StorageClass.INFREQUENT_ACCESS,
          transitionAfter: cdk.Duration.days(30),
        },
        {
          storageClass: s3.StorageClass.GLACIER,
          transitionAfter: cdk.Duration.days(90),
        },
      ],
    },
    // Delete old logs
    {
      prefix: 'logs/',
      expiration: cdk.Duration.days(90),
    },
  ],
});
```

4. API Gateway Optimization

HTTP API vs REST API

```
// HTTP API is 70% cheaper than REST API
// Use when you don't need REST API features

// HTTP API: $1.00/million requests
// REST API: $3.50/million requests
```

Feature	REST API	HTTP API
Cost	$3.50/M	$1.00/M
Lambda integration	Yes	Yes
Request validation	Yes	No
API keys/usage plans	Yes	No
Caching	Yes	No

Feature	REST API	HTTP API

Caching

```
// Enable caching for GET endpoints
const method = resource.addMethod('GET', integration, {
  cacheKeyParameters: ['method.request.querystring.id'],
});

// Cache stage setting
stage.cacheClusterEnabled = true;
stage.cacheClusterSize = '0.5';  // 0.5 GB minimum
```

Note: Cache costs $0.02/hour (0.5 GB). Calculate break-even point.

5. CloudWatch Optimization

Log Retention

```
// Don't keep logs forever
new logs.LogGroup(this, 'LogGroup', {
  retention: logs.RetentionDays.ONE_MONTH,  // Adjust per environment
});
```

Environment	Retention	Reason
Development	7 days	Quick debugging
Staging	14 days	Testing cycles
Production	90 days	Compliance needs

Metric Filters vs. Logs Insights

```
// Use metric filters for known patterns
// Cheaper than running Logs Insights queries repeatedly

new logs.MetricFilter(this, 'ErrorMetric', {
  logGroup,
  metricNamespace: 'Radiant',
  metricName: 'Errors',
  filterPattern: logs.FilterPattern.literal('ERROR'),
});
```

6. ElastiCache Optimization

Reserved Nodes

```
# Purchase reserved nodes for production
aws elasticache purchase-reserved-cache-nodes-offering \
  --reserved-cache-nodes-offering-id xxx
```

Savings: 30-55% for 1-3 year terms

Right-Size Nodes

Use Case	Recommended	Memory
Development	cache.t3.micro	0.5 GB
Small Prod	cache.t3.small	1.4 GB
Medium Prod	cache.r6g.large	13 GB
Large Prod	cache.r6g.xlarge	26 GB

7. Data Transfer Optimization

Use VPC Endpoints

```
// Avoid NAT Gateway costs for AWS services
vpc.addInterfaceEndpoint('S3Endpoint', {
  service: ec2.InterfaceVpcEndpointAwsService.S3,
});

vpc.addInterfaceEndpoint('SecretsManagerEndpoint', {
  service: ec2.InterfaceVpcEndpointAwsService.SECRETS_MANAGER,
});
```

Savings: $0.045/GB saved vs. NAT Gateway

CloudFront for S3

```
// Serve S3 content through CloudFront
// Cheaper data transfer + better performance
const distribution = new cloudfront.Distribution(this, 'CDN', {
  defaultBehavior: {
    origin: new origins.S3Origin(bucket),
  },
});
```

Cost Monitoring

AWS Cost Explorer

```
# Get cost breakdown by service
aws ce get-cost-and-usage \
  --time-period Start=2024-12-01,End=2024-12-31 \
  --granularity MONTHLY \
```

```
  --metrics BlendedCost \
  --group-by Type=DIMENSION,Key=SERVICE
```

CloudWatch Billing Alerts

```typescript
// Alert before surprise bills
new cloudwatch.Alarm(this, 'BillingAlarm', {
  metric: new cloudwatch.Metric({
    namespace: 'AWS/Billing',
    metricName: 'EstimatedCharges',
    dimensionsMap: { Currency: 'USD' },
    statistic: 'Maximum',
    period: cdk.Duration.hours(6),
  }),
  threshold: 1000,  // $1000 threshold
  evaluationPeriods: 1,
});
```

Cost Allocation Tags

```typescript
// Tag all resources for cost tracking
cdk.Tags.of(this).add('Project', 'radiant');
cdk.Tags.of(this).add('Environment', environment);
cdk.Tags.of(this).add('CostCenter', 'platform');
```

Environment-Specific Recommendations

Development

- Use Aurora Serverless v2 (scales to zero)
- Minimal Lambda memory
- No provisioned concurrency
- Short log retention
- Single-AZ deployments

Target: < $100/month

Staging

- Aurora Serverless v2
- Moderate Lambda memory
- No provisioned concurrency
- 14-day log retention
- Single-AZ acceptable

Target: < $300/month

Production

- Aurora Serverless v2 or Reserved (if predictable)
- Right-sized Lambda memory
- Provisioned concurrency for critical paths
- 90-day log retention
- Multi-AZ required

Target: Optimize for reliability, then cost

Monthly Cost Review Checklist

- ☐ Review AWS Cost Explorer for anomalies
- ☐ Check for unused resources (idle RDS, orphan EBS)
- ☐ Review Lambda right-sizing opportunities
- ☐ Check S3 storage class distribution
- ☐ Review data transfer costs
- ☐ Validate reserved capacity utilization
- ☐ Update cost allocation tags
- ☐ Project next month's costs

Tools

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Compute Optimizer
- Lambda Power Tuning
- Infracost - Cost estimation for IaC

PERFORMANCE

RADIANT Performance Guide

Overview

This guide covers performance optimization, caching strategies, and scalability considerations for the RADIANT platform.

Architecture Performance

Request Flow

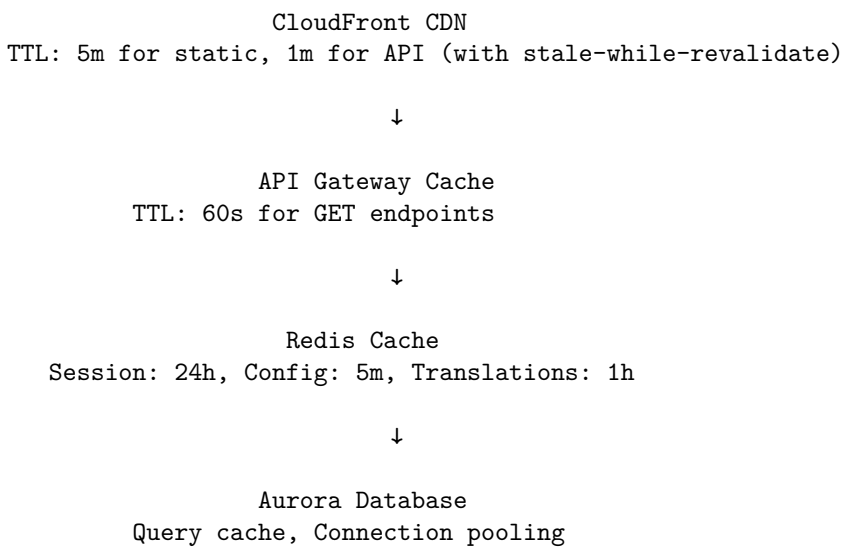
```
Client → CloudFront → WAF → API Gateway → Lambda → Aurora
                                              ↓
                                          Redis Cache
```

Latency Targets

Component	Target	Max Acceptable
CloudFront edge	< 50ms	100ms
WAF processing	< 5ms	20ms
API Gateway	< 20ms	50ms
Lambda cold start	< 500ms	1000ms
Lambda execution	< 200ms	500ms
Database query	< 50ms	200ms
Total P95	**< 500ms**	**2000ms**

Caching Strategy

Multi-Layer Caching



Cache Keys

```
// Session cache
`session:${tenantId}:${userId}` → TTL: 24h

// Configuration cache
`config:${tenantId}:${key}` → TTL: 5m
`config:global:${key}` → TTL: 5m

// Translation cache
```

```
`i18n:${language}:bundle` → TTL: 1h
`i18n:${language}:${key}` → TTL: 1h

// Model cache
`models:${tenantId}:list` → TTL: 5m
`models:${tenantId}:${modelId}` → TTL: 5m

// Rate limit cache
`ratelimit:${tenantId}:${endpoint}` → TTL: 1m
`ratelimit:ip:${ip}` → TTL: 5m
```

Cache Invalidation

```
// Pattern-based invalidation
await redis.del(`config:${tenantId}:*`);

// Event-driven invalidation
eventBridge.putEvents({
  Entries: [{
    Source: 'radiant.config',
    DetailType: 'ConfigUpdated',
    Detail: JSON.stringify({ tenantId, key }),
  }],
});
```

Database Optimization

Connection Pooling

```
// RDS Proxy configuration
const pool = {
  min: 2,
  max: 10,
  idleTimeoutMillis: 30000,
  connectionTimeoutMillis: 5000,
};
```

Query Optimization

```
-- Always use indexes
CREATE INDEX idx_models_tenant_status ON ai_models(tenant_id, status);
CREATE INDEX idx_transactions_tenant_date ON credit_transactions(tenant_id, created_at DESC)

-- Use covering indexes for common queries
CREATE INDEX idx_models_list ON ai_models(tenant_id, status, is_enabled)
  INCLUDE (display_name, category, input_cost_per_1k);
```

```sql
-- Partition large tables by date
CREATE TABLE audit_logs_2024_01 PARTITION OF audit_logs
  FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
```

RLS Performance

```sql
-- Set tenant context once per request
SET app.current_tenant_id = 'tenant-123';

-- All subsequent queries automatically filtered
SELECT * FROM models;  -- Implicitly filtered by RLS
```

Lambda Optimization

Cold Start Reduction

```typescript
// 1. Minimize dependencies
// 2. Use Lambda layers for shared code
// 3. Enable provisioned concurrency for critical functions

// Provisioned concurrency config
new lambda.Function(this, 'Router', {
  // ... config
  provisionedConcurrentExecutions: 10, // Keep 10 warm
});
```

Memory Optimization

```typescript
// Memory vs CPU tradeoff
// More memory = more CPU = faster execution

// Recommended settings by function type:
const memoryConfig = {
  router: 1024,      // Main API - balanced
  billing: 512,      // Light compute
  aiProxy: 2048,     // Heavy compute for AI
  migration: 256,    // Infrequent, light
};
```

Bundling

```typescript
// esbuild configuration for minimal bundle size
{
  bundle: true,
  minify: true,
  treeShaking: true,
  external: ['aws-sdk'], // Use Lambda runtime SDK
```

```
  target: 'node20',
}
```

Rate Limiting

Tier Limits

Tier	RPS	Burst	Daily
Free	10	20	1,000
Starter	50	100	10,000
Professional	100	200	50,000
Business	500	1,000	250,000
Enterprise	2,000	5,000	Unlimited

Implementation

```
// Token bucket algorithm in Redis
const rateLimiter = {
  async checkLimit(tenantId: string, limit: number): Promise<boolean> {
    const key = `ratelimit:${tenantId}`;
    const current = await redis.incr(key);

    if (current === 1) {
      await redis.expire(key, 1); // 1 second window
    }

    return current <= limit;
  }
};
```

Load Testing

Running Tests

```
# Install k6
brew install k6

# Run smoke test
k6 run --env BASE_URL=https://api-dev.radiant.example.com tests/load/k6-config.js

# Run with specific scenario
k6 run --env BASE_URL=https://api-dev.radiant.example.com \
  -e SCENARIO=load tests/load/k6-config.js
```

Performance Baselines

Metric	Baseline	Target
Throughput	500 RPS	2000 RPS
P50 Latency	100ms	50ms
P95 Latency	500ms	200ms
P99 Latency	1000ms	500ms
Error Rate	$< 1\%$	$< 0.1\%$

Scaling

Horizontal Scaling

Component	Scaling Method
Lambda	Automatic (up to account limit)
Aurora	Read replicas, Serverless v2
Redis	ElastiCache cluster mode
API Gateway	Automatic

Vertical Scaling

```
// Aurora Serverless v2 scaling
const database = new rds.DatabaseCluster(this, 'Database', {
  serverlessV2MinCapacity: 0.5,  // Minimum ACUs
  serverlessV2MaxCapacity: 16,   // Maximum ACUs
});

// Lambda memory scaling
const lambda = new lambda.Function(this, 'Function', {
  memorySize: 2048,  // More memory = more CPU
});
```

Monitoring

Key Metrics

```
// CloudWatch metrics to monitor
const metrics = {
  // Latency
  'AWS/ApiGateway/Latency': 'p95 < 500ms',
  'AWS/Lambda/Duration': 'p95 < 200ms',
  'AWS/RDS/ReadLatency': 'avg < 50ms',

  // Throughput
```

```
  'AWS/ApiGateway/Count': 'track trends',
  'AWS/Lambda/Invocations': 'track trends',

  // Errors
  'AWS/ApiGateway/5XXError': 'rate < 1%',
  'AWS/Lambda/Errors': 'rate < 1%',

  // Resources
  'AWS/Lambda/ConcurrentExecutions': '< 80% of limit',
  'AWS/RDS/CPUUtilization': '< 80%',
  'AWS/RDS/DatabaseConnections': '< 80% of max',
};
```

Alerting Thresholds

Metric	Warning	Critical
API P95 Latency	> 1s	> 3s
Error Rate	$> 1\%$	$> 5\%$
Lambda Concurrent	> 500	> 800
DB CPU	$> 70\%$	$> 85\%$
DB Connections	$> 60\%$	$> 80\%$

Best Practices

Do's

- Cache aggressively with proper invalidation
- Use connection pooling
- Minimize cold starts with provisioned concurrency
- Use read replicas for read-heavy workloads
- Implement circuit breakers for external services
- Use async processing for non-critical paths

Don'ts

- Don't make synchronous calls to external APIs in hot paths
- Don't use Lambda for long-running tasks (> 15 min)
- Don't store large objects in Redis
- Don't rely on API Gateway caching for dynamic data
- Don't skip database indexes

Troubleshooting

High Latency

1. Check Lambda cold starts (enable provisioned concurrency)

2. Check database query times (add indexes)
3. Check external API latency (add caching/circuit breaker)
4. Check connection pool exhaustion

High Error Rate

1. Check Lambda errors in CloudWatch Logs
2. Check database connection errors
3. Check rate limiting (429 errors)
4. Check WAF blocked requests

Scaling Issues

1. Check Lambda concurrent execution limit
2. Check database connection limit
3. Check API Gateway throttling
4. Check Redis memory usage

TESTING

RADIANT Testing Guide

Comprehensive guide for testing RADIANT components.

Overview

RADIANT uses a multi-layered testing strategy:

Layer	Tool	Location	Purpose
Unit Tests	Vitest	`**/__tests__/*.test.ts`	Test individual functions/components
Integration Tests	Vitest	`**/__tests__/*.integration.test.ts`	Test service interactions
E2E Tests	Playwright	`apps/admin-dashboard/e2e/`	Test user workflows
Swift Tests	XCTest	`apps/swift-deployer/Tests/`	Test Swift services

Quick Start

```
# Run all tests
pnpm test
```

```
# Run with coverage
pnpm test:coverage

# Run E2E tests
cd apps/admin-dashboard && pnpm test:e2e

# Run Swift tests
cd apps/swift-deployer && swift test
```

Unit Testing

Lambda Handler Tests

Tests for Lambda handlers are located in `__tests__/` directories:

```
packages/infrastructure/lambda/
   admin/
       __tests__/
           handler.test.ts
   billing/
       __tests__/
           handler.test.ts
   shared/
        __tests__/
            auth.test.ts
            errors.test.ts
            services.test.ts
```

Running Lambda Tests

```
cd packages/infrastructure

# Run all Lambda tests
pnpm test

# Run specific handler
pnpm test -- admin
pnpm test -- billing

# Run with coverage
pnpm test:coverage

# Watch mode
pnpm test:watch
```

Writing Lambda Tests

```
import { describe, it, expect, vi, beforeEach } from 'vitest';
import type { APIGatewayProxyEvent, Context } from 'aws-lambda';

// Mock dependencies
vi.mock('../../shared/db', () => ({
  listTenants: vi.fn(),
  getTenantById: vi.fn(),
}));

import { handler } from '../handler';
import { listTenants, getTenantById } from '../../shared/db';

// Create mock context
const mockContext = {
  awsRequestId: 'test-request-id',
  functionName: 'test-handler',
  // ... other required fields
} as Context;

// Create mock event helper
function createMockEvent(overrides = {}): APIGatewayProxyEvent {
  return {
    httpMethod: 'GET',
    path: '/test',
    headers: { Authorization: 'Bearer test-token' },
    // ... default values
    ...overrides,
  };
}

describe('Handler', () => {
  beforeEach(() => {
    vi.clearAllMocks();
  });

  it('should return 200 for valid request', async () => {
    (listTenants as ReturnType<typeof vi.fn>).mockResolvedValue([]);

    const event = createMockEvent({ path: '/admin/tenants' });
    const result = await handler(event, mockContext);

    expect(result.statusCode).toBe(200);
  });
});
```

Shared Module Tests

Test shared utilities and services:

```typescript
// packages/infrastructure/lambda/shared/__tests__/errors.test.ts
import { describe, it, expect } from 'vitest';
import {
  ValidationError,
  NotFoundError,
  isOperationalError,
  toAppError,
} from '../errors';

describe('Error Classes', () => {
  it('should create ValidationError with 400 status', () => {
    const error = new ValidationError('Invalid input');

    expect(error.statusCode).toBe(400);
    expect(error.code).toBe('VALIDATION_ERROR');
  });
});
```

E2E Testing (Admin Dashboard)

Setup

```bash
cd apps/admin-dashboard

# Install Playwright browsers
npx playwright install

# Run E2E tests
pnpm test:e2e

# Run with UI
pnpm test:e2e:ui

# Run specific test file
pnpm test:e2e -- dashboard.spec.ts
```

Test Structure

```
apps/admin-dashboard/e2e/
  dashboard.spec.ts      # Dashboard navigation tests
  deployment.spec.ts     # Deployment workflow tests
  fixtures/
```

```
      test-data.json     # Test fixtures
```

Writing E2E Tests

```typescript
// apps/admin-dashboard/e2e/dashboard.spec.ts
import { test, expect } from '@playwright/test';

test.describe('Dashboard', () => {
  test.beforeEach(async ({ page }) => {
    // Mock authentication
    await page.addInitScript(() => {
      localStorage.setItem('auth_token', 'test-token');
      localStorage.setItem('user', JSON.stringify({
        id: 'test-user',
        email: 'test@example.com',
        role: 'admin',
      }));
    });
  });

  test('should display dashboard home', async ({ page }) => {
    await page.goto('/');
    await expect(page.getByRole('heading', { level: 1 }))
      .toContainText('Dashboard');
  });

  test('should navigate to models page', async ({ page }) => {
    await page.goto('/');
    await page.getByRole('link', { name: 'Models' }).click();
    await expect(page).toHaveURL('/models');
  });
});
```

Swift Testing

Test Structure

```
apps/swift-deployer/Tests/
  RadiantDeployerTests.swift           # Basic unit tests
  RadiantDeployerTests/
      E2ETests/
          DeploymentE2ETests.swift     # E2E workflow tests
      ServiceTests/
          LocalStorageManagerTests.swift
          CredentialServiceTests.swift
```

Running Swift Tests

```
cd apps/swift-deployer

# Run all tests
swift test

# Run specific test class
swift test --filter LocalStorageManagerTests

# Run with verbose output
swift test -v

# Generate coverage (requires llvm-cov)
swift test --enable-code-coverage
```

Writing Swift Tests

```
import XCTest
@testable import RadiantDeployer

final class LocalStorageManagerTests: XCTestCase {
    var storageManager: LocalStorageManager!

    override func setUp() {
        super.setUp()
        storageManager = LocalStorageManager.shared
    }

    override func tearDown() {
        // Cleanup
        super.tearDown()
    }

    func testSaveAndLoadConfiguration() async throws {
        // Given
        let key = "test_config"
        let config = TestConfiguration(name: "Test", value: 42)

        // When
        try await storageManager.save(config, forKey: key)
        let loaded: TestConfiguration? = try await storageManager.load(forKey: key)

        // Then
        XCTAssertNotNil(loaded)
        XCTAssertEqual(loaded?.name, "Test")
```

```
        XCTAssertEqual(loaded?.value, 42)
    }
}
```

Test Utilities

Mock Factories

Use the shared testing utilities:

```typescript
import {
  createMockTenant,
  createMockUser,
  createMockApiKey,
  createMockChatRequest,
  createMockChatResponse,
  createMockApiGatewayEvent,
  createMockLambdaContext,
} from '@radiant/shared/testing';

// Create mock data
const tenant = createMockTenant({ name: 'Test Corp' });
const user = createMockUser({ tenantId: tenant.id });
const event = createMockApiGatewayEvent({
  httpMethod: 'POST',
  body: JSON.stringify({ model: 'gpt-4o' }),
});
```

Assertion Helpers

```typescript
import {
  assertDefined,
  assertEqual,
  assertMatch,
  assertContains,
  assertThrows,
  waitFor,
  sleep,
} from '@radiant/shared/testing';

// Custom assertions
assertDefined(result, 'Result should not be null');
assertMatch(response.id, /^chatcmpl_/, 'Invalid response ID format');

// Wait for async condition
await waitFor(() => service.isReady(), { timeout: 5000 });
```

Mocking Guidelines

Database Mocking

```
vi.mock('../db/client', () => ({
  executeStatement: vi.fn(),
}));

import { executeStatement } from '../db/client';

// Mock return value
(executeStatement as ReturnType<typeof vi.fn>).mockResolvedValue({
  rows: [{ id: '123', name: 'Test' }],
});
```

AWS SDK Mocking

```
vi.mock('@aws-sdk/client-s3', () => ({
  S3Client: vi.fn().mockImplementation(() => ({
    send: vi.fn(),
  })),
  PutObjectCommand: vi.fn(),
}));
```

External API Mocking

```
vi.mock('node-fetch', () => ({
  default: vi.fn(),
}));

import fetch from 'node-fetch';

(fetch as ReturnType<typeof vi.fn>).mockResolvedValue({
  ok: true,
  json: () => Promise.resolve({ data: 'mocked' }),
});
```

CI/CD Integration

Tests run automatically in GitHub Actions:

```
# .github/workflows/ci.yml
jobs:
  test:
```

```yaml
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Setup Node.js
        uses: actions/setup-node@v4
      - name: Install dependencies
        run: pnpm install
      - name: Run tests
        run: pnpm test
        env:
          DATABASE_URL: postgres://test@localhost:5432/test
```

Coverage Requirements

- Minimum 80% coverage for new code
- Critical paths require 90%+ coverage
- All error handling paths must be tested

Best Practices

Do's

- Test behavior, not implementation
- Use descriptive test names
- Mock external dependencies
- Test error cases and edge cases
- Keep tests fast and isolated
- Use factory functions for test data

Don'ts

- Don't test private methods directly
- Don't share state between tests
- Don't test framework code
- Don't ignore flaky tests
- Don't hardcode test data

Test Naming Convention

```javascript
describe('ServiceName', () => {
  describe('methodName', () => {
    it('should return X when given Y', () => {});
    it('should throw error when invalid input', () => {});
    it('should handle empty array gracefully', () => {});
  });
});
```

Debugging Tests

Vitest

```
# Run with verbose output
pnpm test -- --reporter=verbose

# Run single test
pnpm test -- -t "should return 200"

# Debug mode
node --inspect-brk node_modules/.bin/vitest run
```

Playwright

```
# Debug mode with browser
pnpm test:e2e -- --debug

# Generate trace on failure
pnpm test:e2e -- --trace on

# View trace
npx playwright show-trace trace.zip
```

Swift

```
# Run with verbose output
swift test -v

# Run single test
swift test --filter "testSaveAndLoadConfiguration"
```

See Also

- Contributing Guide
- Error Codes Reference
- API Reference

PART 6: API & REFERENCE

RADIANT API Reference

Complete API documentation for the RADIANT AI Platform.

Base URL: `https://api.radiant.example.com/v2`

Authentication: Bearer token (API key or JWT)

```
Authorization: Bearer rad_your_api_key
```

Chat Completions

Create Chat Completion

`POST /v2/chat/completions`

Create a chat completion with any supported model.

Request Body:

Field	Type	Required	Description
`model`	string		Model ID (e.g., `gpt-4o`, `claude-3-sonnet`)
`messages`	array		Array of message objects
`max_tokens`	integer		Maximum tokens to generate
`temperature`	number		Sampling temperature (0-2)
`top_p`	number		Nucleus sampling (0-1)
`stream`	boolean		Stream response tokens
`stop`	string/array		Stop sequences
`functions`	array		Function definitions
`function_call`	string/object		Function calling mode

Message Object:

Field	Type	Required	Description
`role`	string		`system`, `user`, `assistant`, `function`
`content`	string		Message content
`name`	string		Function name (for function messages)

Example Request:

```
{
  "model": "gpt-4o",
  "messages": [
```

```
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello!"}
  ],
  "max_tokens": 1000,
  "temperature": 0.7
}
```

Response:

```
{
  "id": "chatcmpl_abc123",
  "object": "chat.completion",
  "created": 1703980800,
  "model": "gpt-4o",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! How can I help you today?"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 25,
    "completion_tokens": 10,
    "total_tokens": 35
  }
}
```

Models

List Models

`GET /v2/models`

List all available models.

Query Parameters:

Parameter	Type	Description
`category`	string	Filter by category (`chat`, `embedding`, `image`)
`provider`	string	Filter by provider (`openai`, `anthropic`, `google`)

Response:

```
{
  "object": "list",
  "data": [
    {
      "id": "gpt-4o",
      "object": "model",
      "created": 1703980800,
      "owned_by": "openai",
      "display_name": "GPT-4o",
      "category": "chat",
      "context_window": 128000,
      "input_cost_per_1k": 0.005,
      "output_cost_per_1k": 0.015,
      "capabilities": ["chat", "vision", "function_calling"]
    }
  ]
}
```

Get Model

`GET /v2/models/{model_id}`

Get details for a specific model.

Embeddings

Create Embeddings

`POST /v2/embeddings`

Generate embeddings for text.

Request Body:

Field	Type	Required	Description
`model`	string		Embedding model ID
`input`	string/array		Text to embed
`encoding_format`	string		`float` or `base64`

Response:

```
{
  "object": "list",
  "data": [
```

```
    {
      "object": "embedding",
      "index": 0,
      "embedding": [0.0023, -0.0091, ...]
    }
  ],
  "model": "text-embedding-3-small",
  "usage": {
    "prompt_tokens": 8,
    "total_tokens": 8
  }
}
```

Billing

Get Credit Balance

`GET /v2/billing/credits`

Get current credit balance.

Response:

```
{
  "data": {
    "available": 150.50,
    "reserved": 10.00,
    "currency": "USD",
    "updated_at": "2024-01-15T10:30:00Z"
  }
}
```

Get Usage

`GET /v2/billing/usage`

Get usage data for a period.

Query Parameters:

Parameter	Type	Description
`start_date`	string	Start date (YYYY-MM-DD)
`end_date`	string	End date (YYYY-MM-DD)
`group_by`	string	`day`, `model`, `endpoint`

Webhooks

List Webhooks

`GET /v2/webhooks`

List configured webhooks.

Create Webhook

`POST /v2/webhooks`

Request Body:

```
{
  "url": "https://your-server.com/webhook",
  "event_types": ["billing.low_balance", "usage.quota_reached"],
  "description": "Billing alerts"
}
```

Response:

```
{
  "data": {
    "id": "wh_abc123",
    "url": "https://your-server.com/webhook",
    "secret": "whsec_xyz789...",
    "event_types": ["billing.low_balance", "usage.quota_reached"],
    "is_active": true,
    "created_at": "2024-01-15T10:30:00Z"
  }
}
```

Test Webhook

`POST /v2/webhooks/{webhook_id}/test`

Send a test event to the webhook.

Batch Processing

Create Batch Job

`POST /v2/batch/jobs`

Create a batch processing job.

Request Body:

```
{
  "type": "completions",
```

```
  "model": "gpt-4o",
  "input_file": "batch-input.jsonl",
  "options": {
    "system_prompt": "You are a helpful assistant.",
    "max_tokens": 500
  }
}
```

Get Batch Job

`GET /v2/batch/jobs/{job_id}`

Get batch job status and results.

List Batch Jobs

`GET /v2/batch/jobs`

List all batch jobs.

Error Codes

RADIANT uses standardized error codes across all endpoints. See Error Codes Reference for the complete list.

Error Categories

Category	Code Range	Description
Authentication	`RADIANT_AUTH_1xxx`	Token, API key, session errors
Authorization	`RADIANT_AUTHZ_2xxx`	Permission, role, tenant errors
Validation	`RADIANT_VAL_3xxx`	Input validation errors
Resource	`RADIANT_RES_4xxx`	Not found, conflict, quota errors
Rate Limiting	`RADIANT_RATE_5xxx`	Throttling and rate limit errors
AI/Model	`RADIANT_AI_6xxx`	Model, provider, inference errors
Billing	`RADIANT_BILL_7xxx`	Credits, subscription errors
Storage	`RADIANT_STOR_8xxx`	File upload, storage errors
Internal	`RADIANT_INT_9xxx`	Server, database, timeout errors

Common Error Codes

Code	HTTP	Retryable	Description
`RADIANT_AUTH_1001`	401		Invalid authentication token
`RADIANT_AUTH_1004`	401		Invalid API key
`RADIANT_VAL_3001`	400		Required field missing
`RADIANT_RES_4001`	404		Resource not found
`RADIANT_RATE_5001`	429		Rate limit exceeded
`RADIANT_BILL_7001`	402		Insufficient credits
`RADIANT_AI_6004`	502		AI provider error
`RADIANT_INT_9001`	500		Internal server error

Error Response Format:

```
{
  "error": {
    "code": "RADIANT_RATE_5001",
    "message": "Too many requests. Please slow down.",
    "category": "rate_limit",
    "retryable": true,
    "timestamp": "2024-12-25T10:30:00.000Z"
  }
}
```

Retry-After Header: Retryable errors include `Retry-After` header with seconds to wait.

Rate Limits

Tier	Requests/min	Tokens/min
Free	10	10,000
Starter	50	50,000
Professional	100	200,000
Business	500	1,000,000
Enterprise	2,000	Unlimited

Rate limit headers:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1703980860
```

SDKs

TypeScript/JavaScript

```
npm install @radiant/sdk
```

```
import { RadiantClient } from '@radiant/sdk';

const client = new RadiantClient({ apiKey: 'your-key' });
const response = await client.chat.create({
  model: 'gpt-4o',
  messages: [{ role: 'user', content: 'Hello!' }],
});
```

Python

```
pip install radiant-sdk
```

```
from radiant import RadiantClient

client = RadiantClient(api_key="your-key")
response = client.chat.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": "Hello!"}],
)
```

CLI

```
npm install -g @radiant/cli
radiant auth login
radiant chat send "Hello!"
```

Changelog

See CHANGELOG.md for version history.

Support

- **Email:** support@radiant.example.com
- **Documentation:** https://docs.radiant.example.com
- **Status:** https://status.radiant.example.com

API Versioning

RADIANT API Versioning Guide

Overview

This document describes the API versioning strategy for the RADIANT platform.

Versioning Strategy

URL Path Versioning

RADIANT uses URL path versioning as the primary versioning mechanism:

```
https://api.radiant.example.com/v2/models
https://api.radiant.example.com/v2/chat/completions
```

Version Lifecycle

Version	Status	Support End	Deprecation
v1	Deprecated	2024-06-01	Sunset
v2	Current	-	-
v3	Planned	-	Q3 2025

Support Policy

- **Current version**: Full support, all new features
- **Previous version**: Security fixes only, 12 months after new version
- **Deprecated**: No fixes, 6-month sunset warning

Breaking vs Non-Breaking Changes

Non-Breaking Changes (No Version Bump)

These changes can be made to the current version:

- Adding new endpoints
- Adding new optional request parameters
- Adding new response fields
- Adding new enum values (with graceful handling)
- Performance improvements
- Bug fixes that don't change behavior

Breaking Changes (Require New Version)

These changes require a new API version:

- Removing endpoints
- Removing request/response fields
- Changing field types
- Changing validation rules
- Changing authentication methods
- Changing error response formats
- Removing enum values
- Changing default values

Version Header Support

Request Headers

```
# Specify API version via header (optional override)
X-API-Version: 2024-12-01

# Request specific features
X-API-Features: beta-orchestration,streaming-v2
```

Response Headers

```
# Current API version
X-API-Version: 2
X-API-Version-Date: 2024-12-01

# Deprecation warning
Deprecation: true
Sunset: Sat, 01 Jun 2024 00:00:00 GMT
Link: <https://api.radiant.example.com/v2>; rel="successor-version"
```

Deprecation Process

Timeline

```
Day 0:    Announce deprecation
          Add Deprecation header
          Update documentation

Month 3:  Send reminder emails
          Log deprecation warnings

Month 6:  Begin returning 299 status for deprecated endpoints
          Increase warning frequency

Month 12: Sunset - Return 410 Gone
          Redirect to new version docs
```

Deprecation Headers

```typescript
// Add deprecation headers to old endpoints
function addDeprecationHeaders(res: Response, sunset: Date): void {
  res.setHeader('Deprecation', 'true');
  res.setHeader('Sunset', sunset.toUTCString());
  res.setHeader('Link', '<https://api.radiant.example.com/v3>; rel="successor-version"');
}
```

Deprecation Warnings

```typescript
// Log deprecation usage for migration tracking
async function logDeprecatedUsage(req: Request): Promise<void> {
  await analytics.track({
    event: 'deprecated_api_usage',
    properties: {
      endpoint: req.path,
      version: extractVersion(req),
      apiKey: extractKeyId(req),
      tenant: extractTenantId(req),
    },
  });
}
```

Migration Guide Template

v1 to v2 Migration

```markdown
# Migrating from API v1 to v2

## Breaking Changes

### 1. Authentication
- v1: API key in query string (`?api_key=xxx`)
- v2: API key in header (`Authorization: Bearer xxx`)

### 2. Response Format
- v1: Flat response (`{ models: [...] }`)
- v2: Wrapped response (`{ data: [...], meta: {...} }`)

### 3. Error Format
- v1: `{ error: "message" }`
- v2: `{ error: { code: "...", message: "...", details: [...] } }`

## Migration Steps

1. Update authentication headers
```

2. Update response parsing
3. Update error handling
4. Test all endpoints
5. Switch base URL from /v1 to /v2

Feature Flags

Beta Features

```typescript
// Enable beta features via header
const betaFeatures = {
  'beta-orchestration': true,
  'beta-streaming-v2': true,
  'beta-function-calling': true,
};

function isBetaEnabled(req: Request, feature: string): boolean {
  const features = req.headers['x-api-features']?.split(',') || [];
  return features.includes(feature) && betaFeatures[feature];
}
```

Graduated Features

```typescript
// Track feature graduation
const featureGraduation = {
  'function-calling': {
    beta: '2024-06-01',
    stable: '2024-09-01',
    version: 'v2',
  },
  'streaming-v2': {
    beta: '2024-09-01',
    stable: null, // Still in beta
    version: 'v2',
  },
};
```

SDK Versioning

SDK Version Matrix

SDK	Latest	Min API Version	Max API Version
JavaScript	2.5.0	v2	v2
Python	2.3.0	v2	v2
Go	1.2.0	v2	v2
Ruby	1.1.0	v2	v2

SDK Version Headers

```
# SDKs include version info
User-Agent: radiant-js/2.5.0 node/20.10.0
X-Radiant-SDK: js
X-Radiant-SDK-Version: 2.5.0
```

OpenAPI Specification

Versioned Specs

```
/docs/openapi/v2.yaml     # Current version
/docs/openapi/v2-beta.yaml # With beta features
/docs/openapi/v1.yaml     # Deprecated version
```

Schema Versioning

```yaml
# openapi.yaml
openapi: 3.1.0
info:
  title: RADIANT API
  version: 2.0.0
  x-api-version: v2
  x-version-date: '2024-12-01'
  x-deprecation-date: null
```

Testing Versions

Version Compatibility Tests

```typescript
describe('API Version Compatibility', () => {
  it('should support v2 endpoints', async () => {
    const res = await fetch('/v2/health');
    expect(res.status).toBe(200);
  });

  it('should return 410 for sunset v1 endpoints', async () => {
    const res = await fetch('/v1/health');
    expect(res.status).toBe(410);
    expect(res.headers.get('Link')).toContain('/v2');
  });

  it('should include deprecation headers for deprecated endpoints', async () => {
    const res = await fetch('/v2/deprecated-endpoint');
    expect(res.headers.get('Deprecation')).toBe('true');
    expect(res.headers.get('Sunset')).toBeDefined();
  });
});
```

Client Communication

Changelog

Maintain a public changelog:

```
# API Changelog

## 2024-12-24 (v2.5)
- Added: Orchestration patterns endpoint
- Added: Workflow proposals endpoint
- Changed: Increased rate limits for Professional tier

## 2024-12-01 (v2.4)
- Added: AI translation for localization
- Deprecated: Legacy /translate endpoint (sunset 2025-06-01)
```

Email Notifications

```
// Notify developers of breaking changes
async function notifyApiChanges(change: ApiChange): Promise<void> {
  const affectedKeys = await getApiKeysUsingEndpoint(change.endpoint);

  for (const key of affectedKeys) {
    await sendEmail({
      to: key.ownerEmail,
      subject: `RADIANT API: ${change.type} - ${change.endpoint}`,
      template: 'api-change-notification',
      data: {
        change,
        migrationGuide: change.migrationGuideUrl,
        deadline: change.sunsetDate,
      },
    });
  }
}
```

Best Practices

For API Developers

1. **Plan for change**: Design APIs to be extensible
2. **Use optional fields**: Make new fields optional with defaults
3. **Version from day one**: Include version in all endpoints
4. **Document everything**: Keep OpenAPI specs updated
5. **Communicate early**: 12-month deprecation notice minimum

For API Consumers

1. **Pin versions**: Don't use unversioned endpoints
2. **Handle unknown fields**: Ignore unexpected response fields
3. **Monitor deprecation headers**: Set up alerts
4. **Test regularly**: Run integration tests against current version
5. **Subscribe to updates**: Follow changelog and email updates

Contact

Role	Contact	Purpose
API Support	api-support@radiant.example.com	Usage questions
Developer Relations	devrel@radiant.example.com	SDKs, docs
Engineering	engineering@radiant.example.com	Bug reports

Error Codes

RADIANT Error Codes Reference

Standardized error codes for consistent API responses across all RADIANT services.

Overview

All RADIANT errors follow a consistent format:

```
{
  "error": {
    "code": "RADIANT_AUTH_1001",
    "message": "Invalid authentication token. Please sign in again.",
    "category": "authentication",
    "retryable": false,
    "timestamp": "2024-12-25T10:30:00.000Z"
  }
}
```

Error Code Format

`RADIANT_<CATEGORY>_<NUMBER>`

- **RADIANT** - Prefix for all error codes
- **CATEGORY** - Short category identifier (AUTH, VAL, RES, etc.)
- **NUMBER** - Unique 4-digit number within category

Authentication Errors (1xxx)

Errors related to authentication and identity.

Code	HTTP	Retryable	Description
RADIANT_AUTH_1001	401		Invalid authentication token
RADIANT_AUTH_1002	401		Token has expired
RADIANT_AUTH_1003	401		Missing authentication token
RADIANT_AUTH_1004	401		Invalid API key
RADIANT_AUTH_1005	401		API key has expired
RADIANT_AUTH_1006	401		API key has been revoked
RADIANT_AUTH_1007	403		Insufficient API key scope
RADIANT_AUTH_1008	401		Multi-factor authentication required
RADIANT_AUTH_1009	401		Session has expired

Authorization Errors (2xxx)

Errors related to permissions and access control.

Code	HTTP	Retryable	Description
RADIANT_AUTHZ_2001	403		Forbidden - access denied
RADIANT_AUTHZ_2002	403		Tenant ID mismatch
RADIANT_AUTHZ_2003	403		Required role not assigned
RADIANT_AUTHZ_2004	403		Permission denied
RADIANT_AUTHZ_2005	403		Resource access denied
RADIANT_AUTHZ_2006	403		Subscription tier insufficient

Validation Errors (3xxx)

Errors related to input validation.

Code	HTTP	Retryable	Description
RADIANT_VAL_3001	400		Required field is missing
RADIANT_VAL_3002	400		Invalid field format

Code	HTTP	Retryable	Description
`RADIANT_VAL_3003`	400		Value out of allowed range
`RADIANT_VAL_3004`	400		Invalid data type
`RADIANT_VAL_3005`	400		Constraint violation
`RADIANT_VAL_3006`	400		Schema mismatch
`RADIANT_VAL_3007`	400		Invalid JSON in request body
`RADIANT_VAL_3008`	400		Maximum length exceeded
`RADIANT_VAL_3009`	400		Minimum length required

Resource Errors (4xxx)

Errors related to resources and entities.

Code	HTTP	Retryable	Description
`RADIANT_RES_4001`	404		Resource not found
`RADIANT_RES_4002`	409		Resource already exists
`RADIANT_RES_4003`	410		Resource has been deleted
`RADIANT_RES_4004`	423		Resource is locked
`RADIANT_RES_4005`	409		Resource conflict
`RADIANT_RES_4006`	429		Resource quota exceeded

Rate Limiting Errors (5xxx)

Errors related to rate limiting and throttling.

Code	HTTP	Retryable	Description
`RADIANT_RATE_5001`	429		Rate limit exceeded
`RADIANT_RATE_5002`	429		Tenant rate limit exceeded
`RADIANT_RATE_5003`	429		User rate limit exceeded
`RADIANT_RATE_5004`	429		API key rate limit exceeded
`RADIANT_RATE_5005`	429		Model rate limit exceeded
`RADIANT_RATE_5006`	429		Burst limit exceeded

Retry-After Header: Rate limit errors include `Retry-After` header with seconds to wait.

AI/Model Errors (6xxx)

Errors related to AI models and inference.

Code	HTTP	Retryable	Description
RADIANT_AI_6001	404		Model not found
RADIANT_AI_6002	503		Model temporarily unavailable
RADIANT_AI_6003	503		Model overloaded
RADIANT_AI_6004	502		AI provider error
RADIANT_AI_6005	400		Context length exceeded
RADIANT_AI_6006	400		Content filtered by safety
RADIANT_AI_6007	400		Invalid AI request
RADIANT_AI_6008	500		Streaming error
RADIANT_AI_6009	504		AI request timeout
RADIANT_AI_6010	503		Model is cold (warming up)

Billing Errors (7xxx)

Errors related to billing, credits, and subscriptions.

Code	HTTP	Retryable	Description
RADIANT_BILL_7001	402		Insufficient credits
RADIANT_BILL_7002	402		Payment required
RADIANT_BILL_7003	402		Payment failed
RADIANT_BILL_7004	402		Subscription expired
RADIANT_BILL_7005	402		Subscription cancelled
RADIANT_BILL_7006	400		Invalid coupon code
RADIANT_BILL_7007	429		Usage quota exceeded

Storage Errors (8xxx)

Errors related to file storage.

Code	HTTP	Retryable	Description
RADIANT_STOR_8001	413		Storage quota exceeded
RADIANT_STOR_8002	413		File too large
RADIANT_STOR_8003	415		Invalid file type
RADIANT_STOR_8004	500		Upload failed
RADIANT_STOR_8005	404		File not found

Internal Errors (9xxx)

Internal server errors and system failures.

Code	HTTP	Retryable	Description
RADIANT_INT_9001	500		Internal server error
RADIANT_INT_9002	500		Database error
RADIANT_INT_9003	500		Cache error
RADIANT_INT_9004	500		Queue processing error
RADIANT_INT_9005	503		Service unavailable
RADIANT_INT_9006	502		Dependency failure
RADIANT_INT_9007	500		Configuration error
RADIANT_INT_9008	504		Request timeout

Usage in Code

TypeScript/JavaScript

```
import {
  ErrorCodes,
  RadiantError,
  createNotFoundError,
  createValidationError,
  isRetryableError
} from '@radiant/shared';

// Using factory functions (recommended)
throw createNotFoundError('User', userId);
throw createValidationError('Email is required', 'email');

// Direct construction
throw new RadiantError(ErrorCodes.AUTH_INVALID_TOKEN, 'Custom message', {
  details: { tokenPrefix: 'rad_...' },
  requestId: context.awsRequestId,
});

// Check if retryable
if (isRetryableError(error.code)) {
  // Implement retry logic
}
```

Response Format

```
// RadiantError automatically formats responses
const error = new RadiantError(ErrorCodes.RESOURCE_NOT_FOUND);
```

```
return error.toResponse();

// Returns:
// {
//   statusCode: 404,
//   headers: { 'Content-Type': 'application/json' },
//   body: '{"error":{"code":"RADIANT_RES_4001","message":"Resource not found.","category":
// }
```

Client Handling

Retry Logic

```
async function callWithRetry(fn: () => Promise<Response>, maxRetries = 3) {
  for (let i = 0; i < maxRetries; i++) {
    try {
      const response = await fn();
      if (response.ok) return response;

      const error = await response.json();
      if (!error.error.retryable) throw error;

      const retryAfter = response.headers.get('Retry-After') || '5';
      await sleep(parseInt(retryAfter) * 1000);
    } catch (e) {
      if (i === maxRetries - 1) throw e;
    }
  }
}
```

Error Display

```
function getUserMessage(error: RadiantError): string {
  // Error codes include user-friendly messages
  return error.message;
}

function shouldShowRetryButton(error: RadiantError): boolean {
  return error.retryable;
}
```

Adding New Error Codes

1. Add the code to `packages/shared/src/errors/codes.ts`:

```typescript
export const ErrorCodes = {
  // ... existing codes
  MY_NEW_ERROR: 'RADIANT_CAT_NNNN',
} as const;
```

2. Add metadata:

```typescript
export const ErrorCodeMetadata: Record<ErrorCode, {...}> = {
  // ... existing metadata
  [ErrorCodes.MY_NEW_ERROR]: {
    httpStatus: 400,
    category: 'category',
    retryable: false,
    userMessage: 'User-friendly error message.',
  },
};
```

3. Update this documentation.

See Also

- API Reference
- Contributing Guide
- Troubleshooting

Troubleshooting

RADIANT v4.17.0 - Troubleshooting Guide

Common Issues and Solutions

CDK Deployment Failures

Issue	Likely Cause	Solution
Bootstrap failed	Wrong account/region	Verify `aws sts get-caller-identity`
Stack timeout	Slow resource creation	Check CloudFormation events in AWS Console
Resource limit	Service quota exceeded	Request quota increase via AWS Service Quotas
IAM permission denied	Insufficient permissions	Ensure IAM user has AdministratorAccess

Issue	Likely Cause	Solution
Circular dependency	Stack references	Check stack dependencies in CDK code
Asset upload failed	S3 bucket permissions	Verify CDK bootstrap bucket exists

Aurora Database Issues

Issue	Likely Cause	Solution
Connection refused	Security group rules	Verify Lambda SG can reach Aurora SG on port 5432
Authentication failed	Wrong credentials	Check Secrets Manager for correct credentials
Connection timeout	Missing VPC endpoints	Add RDS VPC endpoint to private subnets
Too many connections	Connection exhaustion	Use RDS Proxy or increase max_connections
Slow queries	Missing indexes	Run EXPLAIN ANALYZE and add appropriate indexes
RLS blocking access	Tenant ID not set	Ensure `app.current_tenant_id` is set in session

Lambda Function Errors

Issue	Likely Cause	Solution
Cold start $>$ 10s	VPC attachment	Use provisioned concurrency for critical functions
Timeout	Slow downstream services	Increase timeout, check DB/API latency
Out of memory	Large payloads/responses	Increase memory allocation (also increases CPU)
Permission denied	IAM role misconfigured	Check Lambda execution role policies

Issue	Likely Cause	Solution
Module not found	Missing dependency	Verify all dependencies in package.json
Handler not found	Incorrect handler path	Check function configuration in CDK

LiteLLM / ECS Issues

Issue	Likely Cause	Solution
503 Service Unavailable	ECS task unhealthy	Check ECS service events and task logs
Provider timeout	Invalid API key	Verify provider secrets in Secrets Manager
Rate limited	Too many requests	Implement exponential backoff retry
Wrong model response	Model misconfigured	Check config.yaml model mappings
Container crashes	Memory exhaustion	Increase task memory in CDK
No healthy targets	Health check failing	Verify health check endpoint returns 200

SageMaker Issues (Tier 3+)

Issue	Likely Cause	Solution
Endpoint failed to create	Insufficient capacity	Try different instance type or region
InvocationError	Model loading failed	Check CloudWatch logs for model errors
Slow cold start	Large model size	Use warm pools or smaller model variant
Capacity error	Instance quota reached	Request SageMaker quota increase
Timeout	Long inference time	Increase endpoint timeout or optimize model

Cognito Authentication Issues

Issue	Likely Cause	Solution
Invalid grant	Expired refresh token	Re-authenticate user
User not confirmed	Email not verified	Check email or manually confirm user
MFA required	MFA not set up	Complete MFA setup flow
Invalid client	Wrong client ID	Verify app client ID in configuration
Callback URL mismatch	URL not whitelisted	Add URL to allowed callbacks in Cognito

Admin Dashboard Issues

Issue	Likely Cause	Solution
403 Forbidden	CloudFront OAC issue	Verify S3 bucket policy allows CloudFront
API calls fail	CORS configuration	Check API Gateway CORS settings
Login redirect loop	Cookie domain mismatch	Verify cookie domain matches site domain
Blank page	Build error	Check `next build` output for errors
Slow load	Large bundle size	Enable code splitting and lazy loading

---

Log Locations

Component	CloudWatch Log Group
API Gateway	`/aws/api-gateway/radiant-{env}-api`
Lambda Functions	`/aws/lambda/Radiant-{env}-*`
LiteLLM (ECS)	`/ecs/radiant-{env}-litellm`
SageMaker Endpoints	`/aws/sagemaker/Endpoints/radiant-*`
Aurora PostgreSQL	`/aws/rds/cluster/radiant-{env}/postgresql`
CloudFront	Standard CloudFront logs in S3

Viewing Logs

```
# Tail Lambda logs in real-time
aws logs tail /aws/lambda/Radiant-dev-router --follow
```

```
# View ECS logs
aws logs tail /ecs/radiant-dev-litellm --follow

# Search logs for errors
aws logs filter-log-events \
  --log-group-name /aws/lambda/Radiant-dev-router \
  --filter-pattern "ERROR" \
  --start-time $(date -d '1 hour ago' +%s)000
```

Health Check Endpoints

```
# Platform API health
curl https://api.YOUR_DOMAIN/health
# Expected: {"status":"healthy","version":"4.17.0"}

# LiteLLM health
curl https://api.YOUR_DOMAIN/v2/litellm/health
# Expected: {"status":"healthy"}

# Admin API health
curl https://admin-api.YOUR_DOMAIN/health

# Model registry status
curl https://api.YOUR_DOMAIN/v2/models/status
```

Emergency Procedures

Database Restore from Snapshot

```
# List available snapshots
aws rds describe-db-cluster-snapshots \
  --db-cluster-identifier radiant-prod-cluster \
  --query 'DBClusterSnapshots[*].[DBClusterSnapshotIdentifier,SnapshotCreateTime]' \
  --output table

# Restore from snapshot
aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier radiant-prod-restored \
  --snapshot-identifier your-snapshot-id \
  --engine aurora-postgresql \
  --vpc-security-group-ids sg-xxx \
  --db-subnet-group-name radiant-prod-db-subnet
```

Point-in-Time Recovery

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier radiant-prod-cluster \
  --db-cluster-identifier radiant-prod-recovered \
  --restore-to-time "2024-12-20T10:00:00Z" \
  --vpc-security-group-ids sg-xxx \
  --db-subnet-group-name radiant-prod-db-subnet
```

Rollback CDK Deployment

```
# Rollback to previous deployment
cd packages/infrastructure
npx cdk deploy Radiant-prod-API \
  --context environment=prod \
  --context tier=3 \
  --rollback
```

Disable Problematic Model

```
-- Connect to Aurora and run:
UPDATE models
SET status = 'disabled',
    disabled_reason = 'Emergency disable due to errors',
    updated_at = NOW()
WHERE model_id = 'problematic-model-id';
```

Force Scale Down SageMaker

```
# Scale endpoint to 0 instances
aws sagemaker update-endpoint-weights-and-capacities \
  --endpoint-name radiant-prod-model-endpoint \
  --desired-weights-and-capacities '[{"VariantName":"AllTraffic","DesiredInstanceCount":0}]'
```

Performance Benchmarks

Metric	Target	Acceptable	Action if Exceeded
API Gateway p50 latency	< 50ms	< 100ms	Check Lambda cold starts
API Gateway p99 latency	< 200ms	< 500ms	Enable provisioned concurrency

Metric	Target	Acceptable	Action if Exceeded
Chat streaming start	< 500ms	< 1s	Check LiteLLM/provider latency
Admin dashboard load	< 2s	< 3s	Optimize bundle, enable CDN caching
Model warm-up time	< 3 min	< 5 min	Use larger instance or warm pools
Aurora query latency	< 10ms	< 50ms	Add indexes, optimize queries

---

Support Checklist

When reporting issues, include:

1. **Environment**: dev/staging/prod
2. **Tier**: 1-5
3. **Error message**: Full error text
4. **Request ID**: From response headers
5. **Timestamp**: When the error occurred
6. **Steps to reproduce**: What actions led to the error
7. **Relevant logs**: CloudWatch log excerpts

Useful AWS CLI Commands

```
# Check stack status
aws cloudformation describe-stacks --stack-name Radiant-dev-API \
  --query 'Stacks[0].StackStatus'

# List recent CloudFormation events
aws cloudformation describe-stack-events --stack-name Radiant-dev-API \
  --query 'StackEvents[0:10].[Timestamp,ResourceStatus,ResourceType,LogicalResourceId]' \
  --output table

# Check Lambda function configuration
aws lambda get-function-configuration --function-name Radiant-dev-router

# List ECS services
aws ecs list-services --cluster radiant-dev-cluster
```

```
# Describe ECS service
aws ecs describe-services --cluster radiant-dev-cluster \
  --services radiant-dev-litellm

# Check Secrets Manager secret
aws secretsmanager get-secret-value --secret-id radiant/dev/db-credentials \
  --query 'SecretString' --output text | jq .
```

PART 7: DEPLOYMENT

Deployment Guide

RADIANT v4.18.0 - Deployment Guide

Overview

This guide covers deploying the RADIANT platform from development to production. The platform consists of:

1. **AWS Infrastructure** - CDK stacks for all cloud resources
2. **Admin Dashboard** - Next.js admin interface
3. **Swift Deployer App** - macOS deployment tool with AI assistant

What's New in v4.18.0

- **Unified Package System** - Deploy with atomic component versioning
- **AI Assistant** - Claude-powered deployment guidance in Swift app
- **Configurable Timeouts** - SSM-synced operation timeouts
- **Cost Management** - Real-time cost tracking and alerts
- **Compliance Reports** - SOC2, HIPAA, GDPR, ISO27001 reporting

Prerequisites

AWS Account Setup

Requirement	Action	Verification
AWS Account	Create or use existing	Account ID available
IAM User	Create with AdministratorAccess	Access keys configured

Requirement	Action	Verification
AWS CLI	Install and configure	`aws sts get-caller-identity`
Route 53 Domain (optional)	Register domain	Domain in hosted zone
ACM Certificate (optional)	Request in us-east-1	Certificate validated

Development Environment

Requirement	Version	Verification
Node.js	20.x LTS	`node --version`
pnpm	8.x+	`pnpm --version`
AWS CDK CLI	2.x	`cdk --version`
Xcode	15.x+	`xcode-select -p`
Swift	5.9+	`swift --version`

Quick Start

Automated Deployment

Use the deployment script for a streamlined deployment:

```
# Deploy to dev environment
./scripts/deploy.sh --environment dev

# Deploy to staging
./scripts/deploy.sh --environment staging

# Deploy to production
./scripts/deploy.sh --environment prod
```

> **Note:** RADIANT uses a unified deployment model. All features are available in every deployment. Licensing restrictions are handled at the application level, not infrastructure level.

Verify Deployment

After deployment, verify all resources:

```
./scripts/verify-deployment.sh --environment dev
```

Manual Deployment

1. Install Dependencies

```
cd radiant
npx pnpm install
```

2. Build Shared Package

```
cd packages/shared
npm run build
```

3. Build Lambda Functions

```
cd packages/infrastructure/lambda
npm install --legacy-peer-deps
npm run build
```

4. Build Admin Dashboard

```
cd apps/admin-dashboard
npm install
npm run build
```

CDK Deployment

Bootstrap CDK (One-time per account/region)

```
cd packages/infrastructure
npx cdk bootstrap aws://ACCOUNT_ID/us-east-1 --qualifier radiant
```

Deploy All Stacks

```
# Deploy in order with dependencies
npx cdk deploy --all \
  --context environment=dev \
  --context tier=1 \
  --require-approval never
```

Deploy Individual Stacks

```
# Phase 1: Foundation
npx cdk deploy Radiant-dev-Foundation --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Networking --context environment=dev --context tier=1

# Phase 2: Security & Data
npx cdk deploy Radiant-dev-Security --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Data --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Storage --context environment=dev --context tier=1
```

```
# Phase 3: Auth & AI
npx cdk deploy Radiant-dev-Auth --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-AI --context environment=dev --context tier=1

# Phase 4: API & Admin
npx cdk deploy Radiant-dev-API --context environment=dev --context tier=1
npx cdk deploy Radiant-dev-Admin --context environment=dev --context tier=1
```

Database Migrations

After infrastructure is deployed, run database migrations:

```
# Connect to Aurora and run migrations
cd packages/infrastructure/migrations
./run-migrations.sh --environment dev
```

Migration files are applied in order (44 total): 1. `001_initial_schema.sql` - Base tables 2. `002_tenant_isolation.sql` - RLS policies 3. `003_ai_models.sql` - Providers and models 4. `004_usage_billing.sql` - Usage tracking 5. `005_admin_approval.sql` - Audit logs 6. `006_self_hosted_models.sql` - SageMaker config 7. `007_external_providers.sql` - Provider settings 8. ... (see `migrations/` for full list) 44. `044_cost_experiments_security.sql` - Cost tracking, A/B testing, security

Post-Deployment Configuration

Create First Super Admin

```
aws cognito-idp admin-create-user \
  --user-pool-id YOUR_ADMIN_POOL_ID \
  --username admin@example.com \
  --user-attributes Name=email,Value=admin@example.com \
  --temporary-password TempPass123! \
  --message-action SUPPRESS

aws cognito-idp admin-add-user-to-group \
  --user-pool-id YOUR_ADMIN_POOL_ID \
  --username admin@example.com \
  --group-name super_admin
```

Configure AI Providers

1. Navigate to Admin Dashboard → Providers
2. Add API keys for external providers:
   - OpenAI
   - Anthropic
   - Google AI

- xAI (Grok)
- DeepSeek

3. Verify connectivity with test requests

Environment Configuration

Tiers

Tier	Name	Use Case
1	SEED	Development, testing
2	STARTUP	Small production
3	GROWTH	Medium production
4	SCALE	Large production
5	ENTERPRISE	Enterprise with compliance

Environment Variables

Required environment variables for deployment:

```
export AWS_REGION=us-east-1
export AWS_ACCOUNT_ID=123456789012
export RADIANT_ENVIRONMENT=dev  # dev, staging, prod
export RADIANT_TIER=1           # 1-5
export RADIANT_DOMAIN=example.com
```

Verification

Health Checks

```
# API Health
curl https://YOUR_API_ENDPOINT/health

# Expected response:
# {"status":"healthy","version":"4.18.0"}
```

Smoke Tests

```
# Test chat completions
curl -X POST https://YOUR_API_ENDPOINT/v1/chat/completions \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"model":"gpt-4o-mini","messages":[{"role":"user","content":"Hello"}]}'
```

Troubleshooting

Common Issues

Issue	Cause	Solution
CDK bootstrap fails	Missing permissions	Ensure IAM user has AdministratorAccess
Aurora connection timeout	Security group	Check VPC endpoint and security group rules
Lambda cold starts	Function size	Enable provisioned concurrency for critical functions
Cognito auth fails	Pool configuration	Verify callback URLs and client settings

Logs

```
# View Lambda logs
aws logs tail /aws/lambda/Radiant-dev-router --follow

# View ECS logs (LiteLLM)
aws logs tail /ecs/radiant-dev-litellm --follow
```

Cleanup

To destroy all resources:

```
cd packages/infrastructure
npx cdk destroy --all --context environment=dev --context tier=1
```

Warning: This will delete all data including databases. Export data before destroying.

CI/CD Pipeline

RADIANT includes a GitHub Actions CI/CD pipeline:

Workflow Stages

Stage	Trigger	Actions
Lint	All PRs	ESLint, TypeScript check
Build	All PRs	Build shared, infrastructure, dashboard
Test	All PRs	Unit tests, coverage report
CDK Synth	All PRs	Validate CDK templates
Deploy Dev	Merge to develop	Auto-deploy to dev
Deploy Prod	Merge to main	Deploy with approval

Pre-commit Hooks

Pre-commit hooks run automatically via Husky:

```
# Hooks run on every commit:
- lint-staged (ESLint, Prettier)
- Secret detection
- Version bump enforcement
- Discrete validation
```

To bypass (not recommended):

```
git commit --no-verify -m "message"
```

Manual Deployment from CI

```
# Trigger deployment workflow
gh workflow run deploy.yml -f environment=staging -f tier=2
```

Testing Before Deployment

Always run tests before deploying:

```
# Run all tests
pnpm test

# Run E2E tests
cd apps/admin-dashboard && pnpm test:e2e

# Run CDK synthesis (validates templates)
cd packages/infrastructure && npx cdk synth
```

See Testing Guide for comprehensive testing information.

Support

For issues, check: 1. CloudWatch Logs for error details 2. CDK diff to verify expected changes 3. AWS Console for resource status 4. Troubleshooting Guide 5. Error Codes Reference

---

CDK Stack Dependencies

RADIANT CDK Stack Dependencies

Technical Reference

Version: 4.18.1 | Last Updated: December 2024

This document defines the explicit dependency graph for RADIANT CDK stacks to ensure correct deployment ordering.

Stack Dependency Graph

```
ThermalStack     PerceptionSvc     SageMaker
  (Scaling)        (Vision)       (Self-Host)
```

Stack Definitions

Layer 1: Foundation

Stack	Purpose	Exports
NetworkStack	VPC, Subnets, NAT Gateways	VPC ID, Subnet IDs, Security Group IDs

Layer 2: Core Infrastructure

Stack	Purpose	Dependencies	Exports
DatabaseStack	Aurora PostgreSQL, RDS Proxy	NetworkStack	Cluster ARN, Secret ARN, Proxy Endpoint
SecurityStack	KMS Keys, WAF, GuardDuty	NetworkStack	Key ARNs, WAF ACL ARN
StorageStack	S3 Buckets, CloudFront	NetworkStack	Bucket ARNs, Distribution ID

Layer 3: Authentication

Stack	Purpose	Dependencies	Exports
AuthStack	Cognito User/Identity Pools	Database, Security, Storage	Pool IDs, Client IDs

Layer 4: Application Services

Stack	Purpose	Dependencies	Exports
APIStack	API Gateway, Lambda handlers	Auth, Database, Security	API Endpoint, Lambda ARNs

Stack	Purpose	Dependencies	Exports
AdminStack	Admin API, Dashboard hosting	Auth, Database, Security	Admin API Endpoint
BillingStack	Stripe integration, Usage tracking	Auth, Database	Billing API Endpoint

Layer 5: AI Services

Stack	Purpose	Dependencies	Exports
AIStack	LiteLLM, Model routing	API, Database, Security	AI Gateway Endpoint

Layer 6: Specialized Services

Stack	Purpose	Dependencies	Exports
ThermalStack	Model scaling, State management	AI, Database	Thermal API Endpoint
PerceptionStack	Computer vision pipeline	AI, Storage	Perception API Endpoint
SageMakerStack	Self-hosted model endpoints	AI, Network	Endpoint ARNs

---

CDK Implementation

Explicit Dependencies

```
// packages/infrastructure/lib/main.ts

import { App } from 'aws-cdk-lib';

const app = new App();

// Layer 1
const networkStack = new NetworkStack(app, 'Network', { env });

// Layer 2
```

```
const databaseStack = new DatabaseStack(app, 'Database', {
  env,
  vpc: networkStack.vpc,
});
databaseStack.addDependency(networkStack);

const securityStack = new SecurityStack(app, 'Security', {
  env,
  vpc: networkStack.vpc,
});
securityStack.addDependency(networkStack);

const storageStack = new StorageStack(app, 'Storage', {
  env,
  vpc: networkStack.vpc,
});
storageStack.addDependency(networkStack);

// Layer 3
const authStack = new AuthStack(app, 'Auth', {
  env,
  database: databaseStack,
  security: securityStack,
  storage: storageStack,
});
authStack.addDependency(databaseStack);
authStack.addDependency(securityStack);
authStack.addDependency(storageStack);

// Layer 4
const apiStack = new APIStack(app, 'API', {
  env,
  auth: authStack,
  database: databaseStack,
  security: securityStack,
});
apiStack.addDependency(authStack);

// ... continue for remaining stacks
```

Cross-Stack References

```
// Example: APIStack referencing DatabaseStack exports

export class APIStack extends Stack {
  constructor(scope: Construct, id: string, props: APIStackProps) {
```

```
    super(scope, id, props);

    // Use exports from DatabaseStack
    const clusterArn = props.database.clusterArn;
    const secretArn = props.database.secretArn;

    // Create Lambda with database access
    const handler = new Function(this, 'ApiHandler', {
      environment: {
        AURORA_CLUSTER_ARN: clusterArn,
        AURORA_SECRET_ARN: secretArn,
      },
    });
  }
}
```

Deployment Order

Fresh Install

```
# Deploy in dependency order
cdk deploy NetworkStack
cdk deploy DatabaseStack SecurityStack StorageStack --parallel
cdk deploy AuthStack
cdk deploy APIStack AdminStack BillingStack --parallel
cdk deploy AIStack
cdk deploy ThermalStack PerceptionStack SageMakerStack --parallel
```

Update (with dependencies)

```
# CDK handles ordering automatically when using addDependency
cdk deploy --all
```

Selective Deployment

```
# Deploy specific stack and its dependencies
cdk deploy AIStack --require-approval never
```

Rollback Considerations

Stack	Rollback Safe	Notes
NetworkStack	Caution	May affect all dependent stacks

Stack	Rollback Safe	Notes
DatabaseStack	Caution	Requires DB snapshot for data preservation
SecurityStack	Safe	KMS keys have deletion protection
StorageStack	Caution	S3 buckets may have data
AuthStack	Safe	Cognito pools preserved
APIStack	Safe	Stateless Lambda functions
AdminStack	Safe	Stateless
BillingStack	Caution	May have pending transactions
AIStack	Safe	Stateless routing
ThermalStack	Safe	State in database
SageMakerStack	Caution	May have running endpoints

Validation Script

```
#!/bin/bash
# tools/scripts/validate-stack-deps.sh

echo "Validating CDK stack dependencies..."

# Check for circular dependencies
cdk synth --quiet 2>&1 | grep -i "circular" && {
    echo "  Circular dependency detected!"
    exit 1
}

# Verify deployment order
cdk diff --all 2>&1 | head -50

echo "  Stack dependencies validated"
```

Related Documentation

- Deployment Guide - Full deployment procedures
- Deployer Architecture - Package and deployment flow

PART 8: CHANGELOG

Changelog

All notable changes to RADIANT will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

[4.18.3] - 2024-12-28

Added

Admin Dashboard - Specialty Model Metadata

- **Models Page Enhancement** (`apps/admin-dashboard/app/(dashboard)/models/models-client.ts`
  - Added specialty metadata visibility: hosting type, specialty, capabilities, modalities, license, thermal state
  - Added edit dialog for all specialty metadata fields
  - New summary cards for Self-Hosted vs External model counts
  - New table columns: Hosting, Specialty, Thermal, License, Actions
  - Edit button to modify category, specialty, primary mode, capabilities, modalities, license, commercial use

Provider Rejection Handling & Intelligent Fallback

- **Database Migration** (`packages/infrastructure/migrations/083_provider_rejection_handling.`
  - `provider_rejections` - Track rejections with fallback chain
  - `rejection_patterns` - Learn patterns for smarter fallback selection
  - `user_rejection_notifications` - Notify users of rejected requests
  - `model_rejection_stats` - Per-model rejection statistics
  - Functions: `record_provider_rejection()`, `record_fallback_result()`, `create_rejection_notification()`
- **Rejection Analytics Migration** (`packages/infrastructure/migrations/084_rejection_analytic`
  - `rejection_analytics` - Daily aggregated stats by model/provider/mode/type
  - `rejection_keyword_stats` - Track violation keywords with per-provider counts
  - `rejected_prompt_archive` - Full prompt content for policy review

  - Enhanced `provider_rejections` with prompt_content, orchestration_mode, violation_keywords
  - Views: `rejection_summary_by_provider`, `rejection_summary_by_model`, `top_rejection_keywords`
  - Functions: `record_rejection_with_analytics()`, `get_rejection_analytics_dashboard()`, `flag_keyword_for_review()`
- **Rejection Analytics UI** (`apps/admin-dashboard/app/(dashboard)/analytics/rejections/page.t`
  - Summary cards: Total rejections, fallback success rate, rejected to user, flagged keywords
  - Tabs: By Provider, Violation Keywords, Flagged Prompts, Policy Review
  - View full prompt content for policy investigation
  - Flag keywords for review, add pre-filters
- **Shared Types** (`packages/shared/src/types/provider-rejection.types.ts`)
  - ProviderRejection, RejectionType, FallbackAttempt, RejectionNotification types
  - Constants: REJECTION_TYPE_LABELS, FINAL_STATUS_LABELS
- **Service** (`packages/infrastructure/lambda/shared/services/provider-rejection.service.ts`)
  - `handleRejectionWithFallback()` - Auto-fallback to alternative models
  - `selectFallbackModel()` - Choose model with lowest rejection rate
  - `getUserNotifications()` - Get user's rejection history
  - Integration with AGI Brain Planner
- **Think Tank UI** (`apps/admin-dashboard/components/thinktank/rejection-notifications.tsx`)
  - Bell icon with unread count
  - Sheet panel showing all rejection notifications
  - Suggested actions for users
  - Rejection banners in conversation
- **Documentation** (`docs/PROVIDER-REJECTION-HANDLING.md`)

AI Ethics Standards Framework

- **Database Migration** (`packages/infrastructure/migrations/082_ai_ethics_standards.sql`)
  - `ai_ethics_standards` - Industry AI ethics frameworks with full metadata
  - `ai_ethics_principle_standards` - Maps ethical principles to standard sections
  - Seeded standards: NIST AI RMF 1.0, ISO/IEC 42001:2023, EU AI Act, IEEE 7000, OECD AI Principles, UNESCO AI Ethics
  - View: `ethical_principles_with_standards` - Principles with their standards
  - Functions: `get_principles_with_standards()`, `seed_principle_standard_mappings()`
- **Admin UI** (`apps/admin-dashboard/app/(dashboard)/ethics/page.tsx`)
  - New Standards tab showing all industry frameworks
  - Standards display: name, full name, organization, version, description, URL, mandatory status

  - Principles now show "Derived from / Aligned with" badges linking to standards
  - Color-coded organization types (government, ISO, industry, academic, religious)
- **API Endpoint** (`GET /admin/ethics/standards`)

Windsurf Policies

- **Auto-Build Policy** (`.windsurf/workflows/auto-build.md`)
  - Enforces CHANGELOG.md updates for all features/bug fixes
  - Requires VERSION_HISTORY.json updates on releases
  - Mandates migration header comments for database changes

User Rules System (Memory Rules)

- **Database Migration** (`packages/infrastructure/migrations/080_user_memory_rules.sql`)
  - `user_memory_rules` - User personal AI interaction rules with priority and targeting
  - `preset_user_rules` - Pre-seeded rule templates (20+ presets across 7 categories)
  - `user_rule_application_log` - Tracks when rules are applied to prompts
  - Functions: `get_user_rules_for_preprompt()`, `format_user_rules_for_prompt()`
  - RLS policies for user isolation
- **Memory Categories** (`packages/infrastructure/migrations/081_memory_categories.sql`)
  - `memory_categories` - Hierarchical categorization of memory types
  - 6 top-level categories: Instruction, Preference, Context, Knowledge, Constraint, Goal
  - 14 sub-categories for fine-grained classification
  - Functions: `get_memory_category_tree()`, `get_user_memories_by_category()`
  - Categories: instruction.format, instruction.tone, instruction.source, preference.style, preference.detail, context.personal, context.work, context.project, knowledge.fact, knowledge.definition, knowledge.procedure, constraint.topic, constraint.privacy, constraint.safety, goal.learning, goal.productivity
- **Shared Types** (`packages/shared/src/types/user-rules.types.ts`)
  - UserMemoryRule, PresetUserRule, PresetRuleCategory types
  - MemoryCategory, MemoryCategoryTree, MemoryByCategory types
  - Rule validation function
  - Constants: MEMORY_CATEGORY_LABELS, MEMORY_CATEGORY_ICONS, MEMORY_CATEGORY_COLORS
- **Service** (`packages/infrastructure/lambda/shared/services/user-rules.service.ts`)
  - CRUD operations for user rules
  - Preset rule management
  - `getRulesForPrompt()` - Formats rules for prompt injection
  - `getMemoryCategories()` - Get category tree

  - `getMemoriesByCategory()` - Get memories grouped by category
  - Integration with preprompt-learning.service.ts
- **Think Tank UI** (`apps/admin-dashboard/app/(dashboard)/thinktank/my-rules/`)
  - My Rules tab: View, toggle, edit, delete rules
  - Add from Presets tab: Browse categories, add popular rules
  - Stats: Active rules count, times applied
- **Preset Categories**: Privacy & Safety, Sources & Citations, Response Format, Tone & Style, Accessibility, Topic Preferences, Advanced
- **Documentation** (`docs/USER-RULES-SYSTEM.md`)

Pre-Prompt Learning System

- **Database Migration** (`packages/infrastructure/migrations/079_preprompt_learning.sql`)
  - `preprompt_templates` - Reusable pre-prompt patterns with configurable weights
  - `preprompt_instances` - Tracks actual pre-prompts used in plans with full context
  - `preprompt_feedback` - User feedback with attribution analysis
  - `preprompt_attribution_scores` - Learning data per template/factor combination
  - `preprompt_learning_config` - Admin-configurable learning parameters
  - `preprompt_selection_log` - Selection reasoning audit trail
  - Materialized view for effectiveness summary
  - Functions for score calculation and attribution updates
- **Shared Types** (`packages/shared/src/types/preprompt.types.ts`)
  - Template, Instance, Feedback, Attribution types
  - Selection request/result types
  - Admin dashboard types
- **Service** (`packages/infrastructure/lambda/shared/services/preprompt-learning.service.ts`)
  - Template selection with weighted scoring
  - Variable rendering for dynamic pre-prompts
  - Feedback processing with auto-attribution inference
  - Exploration vs exploitation balancing
  - Admin dashboard data aggregation
- **AGI Brain Integration** - Pre-prompt selection integrated into plan generation
- **Admin Dashboard** (`apps/admin-dashboard/app/(dashboard)/orchestration/preprompts/`)
  - Overview with attribution pie chart and top/low performers
  - Templates tab with usage stats and weight adjustment
  - Attribution analysis with factor breakdown
  - Recent feedback with attribution labels
  - Weight adjustment sliders per template
- **Documentation** (`docs/PREPROMPT-LEARNING-SYSTEM.md`)

SaaS Metrics Dashboard

- **Admin Dashboard** (`apps/admin-dashboard/app/(dashboard)/saas-metrics/`)
  - Comprehensive SaaS business metrics with stunning visualizations
  - Key metrics: MRR, ARR, Gross Margin, Churn Rate, LTV:CAC ratio
  - 5 tabs: Overview, Revenue, Costs, Customers, Models
  - Revenue & Profit trend charts (Area + Line composed)
  - Revenue by Source/Tier pie and bar charts
  - MRR Movement chart (New, Expansion, Churned)
  - Customer growth trends with new/churned breakdown
  - Model profitability table with margin analysis
  - **Excel/CSV Export**: Full metrics report for spreadsheets
  - **JSON Export**: Structured data for integrations
  - Period selection: 7d, 30d, 90d, 12m
- **Documentation** (`docs/SAAS-METRICS-DASHBOARD.md`)
  - Complete feature guide with all metrics definitions
  - Export format documentation
  - API integration details

Revenue Analytics System

- **Types** (`packages/shared/src/types/revenue.types.ts`)
  - Revenue source types: subscription, credit_purchase, ai_markup_external, ai_markup_self_hosted, overage, storage
  - Cost categories: aws_compute, aws_storage, aws_network, aws_database, external_ai, infrastructure, platform_fees
  - Export formats: CSV, JSON, QuickBooks IIF, Xero CSV, Sage CSV
- **Database Migration** (`packages/infrastructure/migrations/078_revenue_analytics.sql`)
  - `revenue_entries` table for individual revenue events
  - `cost_entries` table for infrastructure and provider costs
  - `revenue_daily_aggregates` for pre-computed summaries
  - `model_revenue_tracking` for per-model revenue breakdown
  - `accounting_periods` and `reconciliation_entries` for month-end close
  - Auto-aggregation triggers for daily summaries
- **Revenue Service** (`packages/infrastructure/lambda/shared/services/revenue.service.ts`)
  - Dashboard with gross revenue, COGS, gross profit, and margin calculations
  - Revenue breakdown by source, tenant, product, and model
  - Multi-format export: CSV summary, JSON details, QuickBooks IIF, Xero CSV, Sage CSV
- **Admin Dashboard** (`apps/admin-dashboard/app/(dashboard)/revenue/`)
  - Revenue Analytics page with period selection (7d, 30d, 90d, YTD, 12m)
  - Summary cards: Gross Revenue, Total COGS, Gross Profit, Gross Margin
  - Revenue breakdown by source with visual bars

  - Cost breakdown by AWS service and external providers
  - Revenue by model with provider cost vs customer charge
  - Revenue by tenant rankings
  - Export dropdown for all accounting formats

Documentation

Think Tank Easter Eggs Guide

- **New Documentation** (`docs/THINK-TANK-EASTER-EGGS.md`)
  - Complete guide to all 10 easter eggs with activation commands
  - Deactivation methods: toggle, `/normal`, timeout, settings
  - Available easter eggs: Konami Code, Chaos Mode, Socratic Mode, Victorian, Pirate, Haiku, Matrix, Disco, Dad Jokes, Emissions
  - Achievement integration for easter egg discovery
  - Admin-only configuration notes (easter eggs are Think Tank consumer feature only)
  - API reference for triggering and deactivating easter eggs

---

[4.18.2] - 2024-12-28

Added

Think Tank Delight System

- **Core Service** (`packages/infrastructure/lambda/shared/services/delight.service.ts`)
  - Personality modes: professional, subtle, expressive, playful
  - 9 trigger types: domain_loading, time_aware, model_dynamics, etc.
  - 3 injection points: pre_execution, during_execution, post_execution
  - Achievement tracking with 13 predefined achievements
  - Easter eggs with 10 hidden features
  - Sound themes: default, mission_control, library, workshop, emissions
- **AGI Brain Integration** (`delight-orchestration.service.ts`)
  - Real-time delight messages during workflow execution
  - Step-specific contextual messages for all 11 step types
  - Orchestration mode-specific personality
- **Real-time Events** (`delight-events.service.ts`)
  - EventEmitter for streaming delight messages
  - SSE stream support for client consumption
  - Plan and step update notifications
- **Persistent Statistics** (`migrations/076_delight_statistics.sql`)
  - Daily statistics aggregation with automatic triggers
  - Message performance tracking
  - Achievement unlock analytics
  - Easter egg discovery metrics

  - User engagement leaderboards
  - 12-week trend analysis
- **Admin Dashboard**
  - Delight management UI (`app/(dashboard)/thinktank/delight/page.tsx`)
  - Statistics dashboard (`delight/statistics/page.tsx`)
  - Category management, message CRUD, analytics

Localization System

- **Database Migration** (`migrations/074_localization_registry.sql`)
  - UI string registry with namespace support
  - Translation storage for multiple languages
  - Seeded with initial English strings
- **Translation Hook** (`hooks/useTranslation.ts`)
  - React hook for accessing translations
  - Language switching support
  - RTL language detection
- **Language Settings**
  - Language selector in Think Tank Settings
  - API route for fetching translations

Windsurf Workflows

- **Policy Workflows** (`.windsurf/workflows/`)
  - `no-hardcoded-ui-text.md` - Localization enforcement policy
  - `no-mock-data.md` - Production code policy
  - `no-stubs.md` - No stubs in production
  - `hipaa-phi-sanitization.md` - HIPAA compliance policy

Changed

Unified Deployment Model

- Removed tier 1-5 deployment selection from Swift Deployer
- Single deployment model with all features available
- Licensing restrictions handled at application level, not infrastructure
- Updated `CDKService.deploy()` to remove tier parameter
- Simplified `ParameterEditorView` and `DeployView`

Documentation

- Updated `DEPLOYMENT-GUIDE.md` with unified deployment model
- Added Delight System section to `THINK-TANK-USER-GUIDE.md`
- Added Section 20 to `RADIANT-ADMIN-GUIDE.md` for Delight administration

[4.18.1] - 2024-12-25

Added

Standardized Error Handling System

- **Error Codes Module** (`packages/shared/src/errors/`)
  - 60+ standardized error codes with format `RADIANT_<CATEGORY>_<NUMBER>`
  - `RadiantError` class with automatic HTTP response formatting
  - Factory functions: `createNotFoundError`, `createValidationError`, etc.
  - Error metadata including `retryable` flag and user-friendly messages
  - Full documentation in `docs/ERROR_CODES.md`

Comprehensive Test Coverage

- **Lambda Handler Tests** (`packages/infrastructure/lambda/*/__tests__/`)
  - Admin handler tests: routes, authorization, error handling
  - Billing handler tests: subscriptions, credits, transactions
  - Auth module tests: token validation, permissions, tenant access
  - Error module tests: all error classes and utilities
- **Swift Service Tests** (`apps/swift-deployer/Tests/`)
  - `LocalStorageManagerTests`: configuration storage, deployment history
  - `CredentialServiceTests`: credential validation, secure storage

Documentation

- **Testing Guide** (`docs/TESTING.md`) - Comprehensive testing documentation
- **Error Codes Reference** (`docs/ERROR_CODES.md`) - Full error code listing

Changed

Code Quality Improvements

- **Type Safety**: Replaced `any` casts with proper interfaces in `cost/page.tsx`
- **Service Consolidation**: Removed duplicate `SchedulerService` (kept canonical version in `shared/services/`)
- **Pre-commit Hooks**: Added `lint-staged` configuration with ESLint, Prettier, SwiftFormat

Fixed

TypeScript Errors

- `db/client.ts`: Fixed AWS SDK Field union type narrowing issue

- `error-logger.ts`: Fixed SqlParameter type inference
- `localization.ts`: Fixed Map iterator compatibility
- `result-merging.ts`: Fixed Set spread iterator issue
- `voice-video.ts`: Fixed Buffer to Blob conversion

Documentation Updates

- Updated README.md with project structure, testing, and CI/CD info
- Updated CONTRIBUTING.md with error handling and testing guidelines
- Updated API_REFERENCE.md with standardized error codes
- Updated DEPLOYMENT-GUIDE.md with CI/CD pipeline info

[4.18.0] - 2024-12-24

Added

PROMPT-33 Update v3 - Unified Deployment System

Package System

- Unified package format (.pkg) with atomic component versioning
- `manifest.json` schema v2.0 with component checksums
- `VERSION_HISTORY.json` for rollback chain support
- Independent Radiant/Think Tank versioning with `touched` flag detection
- Package build scripts (`tools/scripts/build-package.sh`)

Build System & Version Control

- `VERSION`, `RADIANT_VERSION`, `THINKTANK_VERSION` files in repo root
- Husky `commit-msg` hook for Conventional Commit validation
- Enhanced `pre-commit` hook with version bump enforcement
- `bump-version.sh` for automated version management
- `generate-changelog.sh` for changelog automation
- `validate-discrete.sh` and `validate-discrete-ast.sh` for component isolation

Swift Deployer Enhancements

- **AIAssistantService**: Claude API integration with Keychain storage
- **LocalStorageManager**: SQLCipher encrypted local storage
- **TimeoutService**: Configurable operation timeouts with SSM sync
- Connection monitoring with 60-second polling
- Fallback behavior when AI unavailable

Cost Management (Admin Dashboard)

- **CostAnalytics** component with trend charts and model breakdown
- **InsightCard** component for AI recommendations (requires human approval)
- Cost alerts for budget thresholds and spike detection
- Product segmentation (Radiant/Think Tank/Combined)
- Neural Engine cost optimization suggestions

Compliance Reports

- **CustomReportBuilder** for configurable compliance reports
- SOC2, HIPAA, GDPR, ISO27001 framework support
- Custom metric selection and filtering
- Scheduled report generation with email delivery
- PDF, CSV, JSON export formats

Security & Intrusion Detection

- Security dashboard with anomaly detection
- Geographic anomaly detection (impossible travel)
- Session hijacking detection
- Failed login monitoring and alerts
- **anomaly-detector** Lambda function

A/B Testing Framework

- **ExperimentDashboard** for experiment management
- Hash-based sticky variant assignment
- Statistical analysis (t-test, chi-square, p-value)
- **experiment-tracker** Lambda function

Deployment Settings

- **DeploymentSettings** component with SSM sync
- Lock-step mode for component versioning
- Max version drift configuration
- Automatic rollback on failure
- **OperationTimeouts** component for all deployment operations

Database Schema

- Migration 044: cost_events, cost_daily_aggregates, cost_alerts
- Migration 044: experiments, experiment_assignments, experiment_metrics
- Migration 044: security_anomalies, compliance_reports
- Migration 044: deployment_timeouts, deployment_settings

Changed

- Updated all version constants from 4.17.0 to 4.18.0
- Enhanced Settings page with Deployment and Timeouts tabs
- Added CustomReportBuilder to Compliance page
- Integrated InsightsList into CostAnalytics component

[4.17.0] - 2024-12-24

Added

Infrastructure

- 36 database migrations covering all platform features
- 9 CDK stacks for AWS deployment
- Docker Compose for local development
- LocalStack integration for AWS service emulation

Lambda Services

- Billing service with 7-tier subscription model
- Storage billing with tiered pricing
- Localization service with AI translation
- Configuration management with tenant overrides
- Migration approval with dual-admin workflow
- Neural orchestration patterns
- Feedback learning system
- Workflow proposals

Admin Dashboard

- 14 fully functional pages
- Models management
- Providers management with health monitoring
- Billing & credits dashboard
- Storage usage monitoring
- Localization management
- Configuration editor
- Migration approval workflow
- Audit logs viewer
- Notifications center
- User settings

Developer Experience

- GitHub Actions CI/CD pipelines
- Dependabot configuration

- Pre-commit hooks with secret detection
- OpenAPI 3.1 specification
- Playwright E2E tests
- Vitest unit tests
- Comprehensive documentation

Security

- Row-level security (RLS) on all tenant tables
- Dual-admin approval for production migrations
- MFA support for administrators
- Secret scanning in pre-commit hooks

[4.16.0] - 2024-12-01

Added

- Initial Swift Deployer app structure
- Base CDK infrastructure
- Core database schema

[4.15.0] - 2024-11-15

Added

- Project initialization
- Monorepo structure with pnpm workspaces

Version History

Version	Date	Highlights
4.18.0	2024-12-24	PROMPT-33: Unified Deployment System, Cost Management, Compliance, A/B Testing
4.17.0	2024-12-24	Full platform implementation
4.16.0	2024-12-01	Swift Deployer, base CDK
4.15.0	2024-11-15	Project initialization