Contents

RADIANT v4.17.0 - PROMPT 32: AI-OPTIMIZED FOR CODE GENERATION

Complete Implementation Prompt for Windsurf/Claude Opus 4.5
Version: 4.17.0 | December 2024 | ~2.2MB | 400-500 AI-assisted hours Status: AI-OPTIMIZED - Enhanced for reliable Swift & TypeScript code generation

---

AI CODE GENERATION ENHANCEMENTS (v4.17.0)

Enhancement	Description
Complete Swift App Entry Point	Fixed missing RadiantDeployerApp.swift with proper @main struct
Xcode Project Structure	Added Package.swift, Info.plist, entitlements templates
Version Constant	Single `RADIANT_VERSION` constant replaces hardcoded strings
File Creation Order	Explicit dependency graph for AI implementation
Preview Providers	Added SwiftUI previews for all Views
Platform Requirements	Clear macOS 13.0+, Swift 5.9+, Xcode 15+ markers
Sendable Conformance	All types crossing actor boundaries are Sendable
SQLCipher Setup	Complete SPM dependency configuration
AI Implementation Notes	MARK comments explaining context and dependencies

---

WHAT'S NEW IN v4.16.0 (PROMPT 30)

Version	Section	Feature	Description
v4.13.0	43	**Billing & Credits System**	7-tier subscriptions (FREE→ENTERPRISE PLUS), prepaid credits ($10=1 credit), volume discounts, Stripe integration, credit pools for families/teams
v4.14.0	44	**Storage Billing System**	Tiered S3/DB/backup pricing, quota management, overage billing, usage tracking per tenant

Version	Section	Feature	Description
v4.15.0	45	**Versioned Subscriptions & Grandfathering**	Plan version snapshots, locked pricing for existing subscribers, migration offers with incentives
v4.16.0	46	**Dual-Admin Migration Approval**	Two-person approval for production migrations, self-approval prevention, configurable policies, complete audit trail

New Database Tables (v4.13-v4.16):

- `subscription_tiers`, `subscription_add_ons`, `credit_pools`, `credit_pool_members`
- `credit_transactions`, `credit_purchases`, `subscriptions`, `auto_purchase_settings`
- `credit_usage`, `billing_events`
- `storage_usage`, `storage_pricing`, `storage_events`
- `subscription_plan_versions`, `grandfathered_subscriptions`, `plan_change_audit`
- `migration_approval_requests`, `migration_approvals`, `migration_approval_policies`

Total Platform Stats (v4.16.0):

- **46 sections** across 9 implementation phases
- **140+ database tables** with RLS
- **106+ AI models** (50 external + 56 self-hosted)
- **7-tier subscription model** with prepaid credits
- **18 languages** with AI translation
- **12 configuration categories** with hot reload

CRITICAL: READ THIS FIRST - WHAT THIS PROMPT BUILDS

TWO DISTINCT COMPONENTS

This prompt creates **TWO SEPARATE THINGS** that must be understood before implementation:

```
RADIANT PLATFORM ARCHITECTURE


    COMPONENT 1: SWIFT DEPLOYER              ← Built ONCE on developer Mac
    (Section 1 - ~2,400 lines)
```

- Native macOS application (SwiftUI)

```
• Manages AWS credentials securely
• Orchestrates CDK deployments
• Runs database migrations
• Monitors deployment progress
• One-click infrastructure management

THE SWIFT APP IS THE TOOL THAT DEPLOYS
RADIANT. USERS NEVER TOUCH A TERMINAL.


                        deploys


   COMPONENT 2: AWS INFRASTRUCTURE        ← Deployed TO AWS (repeatedly)
   (Sections 0, 2-46 - ~56,000 lines)

• CDK stacks (VPC, Cognito, Aurora,
  Lambda, API Gateway, SageMaker, etc.)
• 50+ Lambda functions
• 140+ database tables with RLS
• Admin Dashboard (Next.js)
• Think Tank consumer platform
• 106+ AI model integrations
• Multi-tenant, multi-app isolation

THIS IS THE "PAYLOAD" - THE ACTUAL
RADIANT PLATFORM RUNNING ON AWS.
```

Key Understanding

Component	Where It Runs	When Built	Purpose
Swift Deployer (Section 1)	Developer's Mac	Once	Deploys and manages AWS infrastructure
AWS Payload (Sections 0, 2-46)	AWS Cloud	Per environment (dev/staging/prod)	The actual RADIANT platform

The Swift app is built ONCE locally. It then deploys the AWS infrastructure REPEATEDLY to different environments.

IMPLEMENTATION STRATEGY FOR CLAUDE/WINDSURF

How to Use This Prompt

This document is ~2MB and exceeds single-context limits. **DO NOT attempt to implement everything at once.**

Recommended Approach: Phase-by-Phase Implementation

```
IMPLEMENTATION PHASES (Follow This Order)

PHASE 1: Foundation (Implement First)

• Section 0: Shared Types & Constants (~1,400 lines)
• Section 1: Swift Deployment App (~2,400 lines)
• Section 2: CDK Infrastructure Stacks (~2,700 lines)

PHASE 2: Core Infrastructure

• Section 3: CDK AI & API Stacks (~2,900 lines)
• Section 4: Lambda Functions - Core (~3,900 lines)
• Section 5: Lambda Functions - Admin & Billing (~1,700 lines)
• Section 6: Self-Hosted Models (~1,600 lines)
• Section 7: Database Schema (~3,500 lines)

PHASE 3: Admin & Deployment

• Section 8: Admin Web Dashboard (~4,200 lines)
• Section 9: Assembly & Deployment Guide (~900 lines)

PHASE 4: AI Features

• Sections 10-17: Visual AI, Brain, Analytics, Neural Engine, etc.
• (~1,500 lines total - smaller sections)

PHASE 5: Consumer Platform

• Sections 18-28: Think Tank, Concurrent Chat, Collaboration, etc.
• (~2,700 lines total)

PHASE 6: Advanced Features

• Sections 29-35: Provider Registry, Time Machine, Orchestration, etc.
• (~6,200 lines total)

PHASE 7: Intelligence Layer
```

- Sections 36-39: Unified Registry, Feedback Learning, Neural Orchestration
- (~5,500 lines total)

PHASE 8: Platform Hardening

- Section 40: Application-Level Data Isolation (~1,800 lines)
- Section 41: Complete Internationalization (~3,100 lines)
- Section 42: Dynamic Configuration Management (~2,200 lines)

Instructions for AI Implementation

When implementing this prompt, follow these rules:

1. **Implement ONE PHASE at a time** - Complete all sections in a phase before moving to the next
2. **Follow the dependency graph** - Sections depend on previous sections (see detailed graph below)
3. **Section 0 FIRST, always** - Shared types must exist before anything else
4. **Test each phase** - Verify deployment works before adding more complexity
5. **Reference, don't duplicate** - Import from `@radiant/shared`, never redefine types

Sample Implementation Commands

```
# Phase 1 implementation prompt:
"Implement RADIANT Phase 1: Sections 0, 1, and 2.
 Start with Section 0 (shared types), then Section 1 (Swift app),
 then Section 2 (CDK infrastructure). Follow all specifications exactly."

# Phase 2 implementation prompt:
"Implement RADIANT Phase 2: Sections 3-7.
 Phase 1 is complete. Now add CDK AI stacks, Lambda functions,
 self-hosted models, and database schema."

# Continue for each phase...
```

INTEGRATION NOTES (Inherited from v4.9.0)

This prompt has been fully integrated with the following key changes:

1. **AuthContext Unified** - Single source of truth in Section 4, all duplicates removed
2. **Migration Numbering Fixed** - Sequential 001-050 with clear section mapping
3. **Version Tags Complete** - All sections now have version tags
4. **Database Connection Standardized** - All Lambdas use `DATABASE_URL` pattern
5. **RLS Policies Consistent** - All use `app.current_tenant_id`
6. **Duplicate Types Removed** - Single definitions in Section 0
7. **Leftover Markers Cleaned** - No more "END OF PROMPT X" within document

8. **TODO Items Addressed** - Resolved or marked with clear implementation notes
9. **Dependency Order Verified** - Sections ordered by implementation dependencies
10. **Think Tank Branding** - All "Radiant Solver" references updated to "Think Tank"

STRUCTURAL FIXES (Inherited from v4.9.0)

Issue #1: Migration Number Conflicts (FIXED)

Problem: Migrations used inconsistent numbering (001-007, then jumped to 020-042) **Solution**: Renumbered all migrations sequentially: - 001-007: Core schema (Section 7) - 008-009: Reserved - 010-019: Infrastructure features (Sections 10-17) - 020-029: Consumer features (Sections 18-27) - 030-039: Advanced features (Sections 28-37) - 040-046: Latest features (Sections 38-46)

Issue #2: Duplicate Type Definitions (FIXED)

Problem: `AuthContext`, `ModelPricing`, `ExternalProvider` defined multiple times **Solution**: Single source of truth in Section 0 (`packages/shared/src/types/`)

Issue #3: Missing Version Tags (FIXED)

Problem: Sections 32-35 and 40 had no version tags **Solution**: Added: - Section 32-33: v4.0.0 (Time Machine) - Section 34-35: v4.1.0 (Orchestration Engine) - Section 40: v4.6.0 (App Isolation)

Issue #4: Database Connection Inconsistency (FIXED)

Problem: Mixed `DATABASE_URL` and `DB_HOST/DB_NAME/DB_USER/DB_PASSWORD` **Solution**: Standardized all Lambda functions to use `DATABASE_URL`:

```
const pool = new Pool({ connectionString: process.env.DATABASE_URL });
```

Issue #5: RLS Policy Variable Mismatch (FIXED)

Problem: Some policies used `app.current_tenant` instead of `app.current_tenant_id` **Solution**: All RLS policies now consistently use `current_setting('app.current_tenant_id')::UUID`

Issue #6: Missing Version v3.4.0 (FIXED)

Problem: Version history jumped from v3.3.0 to v3.5.0 **Solution**: Added v3.4.0 as reserved/internal milestone (no user-facing features)

Issue #7: Leftover Prompt Markers (FIXED)

Problem: "END OF PROMPT 15" and "END OF PROMPT 16" embedded within document **Solution**: Removed all internal prompt markers, only final END marker remains

---

COMPLETE SECTION REFERENCE

Section	Name	Lines	Phase	Version	Key Deliverables
0	Shared Types & Constants	~1,400	1	v2.0.0	TypeScript types, tier configs, regions
1	Swift Deployment App	~2,400	1	v2.0.0	macOS GUI, credential management, deployment orchestration
2	CDK Infrastructure Stacks	~2,700	1	v2.0.0	VPC, Aurora, Cognito, S3, base Lambda
3	CDK AI & API Stacks	~2,900	2	v2.1.0	API Gateway, SageMaker, LiteLLM
4	Lambda Core	~3,900	2	v2.1.0	Router, chat, model handlers, auth context
5	Lambda Admin & Billing	~1,700	2	v2.1.0	Tenant CRUD, billing, invoicing
6	Self-Hosted Models	~1,600	2	v2.2.0	SageMaker endpoints, thermal management
7	Database Schema	~3,500	2	v2.2.0	140+ tables, RLS policies, migrations
8	Admin Dashboard	~4,200	3	v2.2.0	Next.js admin UI, all management pages
9	Assembly & Deployment	~900	3	v2.2.0	Build scripts, deployment guide
10	Visual AI Pipeline	~230	4	v2.3.0	Image processing orchestration
11	RADIANT Brain	~280	4	v2.4.0	Intelligent routing engine
12	Metrics & Analytics	~160	4	v2.5.0	Usage tracking, dashboards
13	Neural Engine	~230	4	v3.0.0	User preferences, pgvector embeddings
14	Error Logging	~200	4	v3.1.0	Centralized error capture
15	Credentials Registry	~220	4	v3.2.0	External API key management

Section	Name	Lines	Phase	Version	Key Deliverables
16	AWS Admin Credentials	~90	4	v3.2.0	AWS credential integration
17	Auto-Resolve API	~140	4	v3.3.0	Automatic request handling
18	Think Tank Platform	~300	5	v3.5.0	Consumer AI interface
19	Concurrent Chat	~460	5	v3.6.0	Split-pane multi-chat UI
20	Real-Time Collaboration	~180	5	v3.6.0	Yjs CRDT multi-user editing
21	Voice & Video	~220	5	v3.6.0	Audio input/output integration
22	Persistent Memory	~200	5	v3.6.0	Cross-session user memory
23	Canvas & Artifacts	~210	5	v3.6.0	Rich content editing
24	Result Merging	~230	5	v3.6.0	AI response synthesis
25	Focus Modes & Personas	~210	5	v3.6.0	Domain presets, custom AI personalities
26	Scheduled Prompts	~250	5	v3.6.0	Recurring automated tasks
27	Family & Team Plans	~250	5	v3.6.0	Shared subscriptions
28	Analytics Integration	~200	5	v3.6.0	Think Tank usage analytics
29	Admin Extensions	~430	6	v3.7.0	Additional admin pages
30	Dynamic Provider Registry	~540	6	v3.7.0	Database-driven providers, xAI/Grok
31	Model Selection & Pricing	~1,970	6	v3.8.0	User model choice, editable pricing

Section	Name	Lines	Phase	Version	Key Deliverables
32	Time Machine Core	~1,880	6	v4.0.0	Chat history versioning
33	Time Machine UI	~1,320	6	v4.0.0	Visual timeline interface
34	Orchestration Engine	~680	6	v4.1.0	Database-driven workflows
35	License Management	~340	6	v4.1.0	AI model license tracking
36	Unified Model Registry	~1,400	7	v4.2.0	106+ models, sync service
37	Feedback Learning	~1,130	7	v4.3.0	Explicit/implicit/voice feedback
38	Neural Orchestration	~900	7	v4.4.0	Neural-first architecture
39	Workflow Proposals	~4,600	7	v4.5.0	Evidence-based workflow generation
40	App Isolation	~1,800	8	v4.6.0	Per-app data separation
41	Internationalization	~2,160	8	v4.7.0	18 languages, AI translation
42	Dynamic Configuration	~2,200	8	v4.8.0	Runtime parameter management
43	Billing & Credits System	~1,900	9	v4.13.0	7-tier subscriptions, prepaid credits, Stripe
44	Storage Billing	~700	9	v4.14.0	Tiered S3/DB pricing, quotas
45	Versioned Subscriptions	~750	9	v4.15.0	Grandfathering, migration offers
46	Dual-Admin Approval	~700	9	v4.16.0	Two-person migration approval

---

Also includes all v4.8.0 features:

Feature	Description
Localization Registry	All UI strings stored in database - zero hardcoded text allowed
18 Supported Languages	EN, ES, FR, DE, PT, JA, KO, ZH-CN, ZH-TW, AR, IT, NL, RU, PL, TR, VI, TH, HI
AI Auto-Translation	AWS Bedrock (Claude) translates new strings, flags for admin review
Translation Admin UI	Browse, edit, approve translations in Admin Dashboard
Hardcode Prevention	ESLint rules block hardcoded strings at build time
React i18n Hooks	`useTranslation()` hook for web apps with fallback chain
Swift Localization	Native Swift localization service for Think Tank app
Translation Alerts	Admins notified when AI translations need review
Translation Coverage	Dashboard showing % translated per language
Real-Time Sync	Translation updates propagate instantly via WebSocket

Section 41: Complete Internationalization System

Why This Matters: - **Global Reach**: Support users in their native language across 18 languages - **Quality Control**: AI translations flagged for human review before production - **Developer Experience**: ESLint catches hardcoded strings at build time - **Maintainability**: Single source of truth in database, not scattered across code - **Compliance**: GDPR/accessibility requirements for localized content

Key Components: - `localization_registry` - Master list of all translatable strings with keys - `localization_translations` - Per-language translations with status tracking - `localization_languages` - Supported languages with metadata - Translation Lambda - Auto-translates via Bedrock when new strings added - Admin UI - Full CRUD for translations with approval workflow - ESLint Plugin - Prevents hardcoded strings in TypeScript/React - Swift Localization Service - Native i18n for Think Tank iOS/macOS app

---

Also includes all v4.6.0 features:

WHAT'S NEW IN v4.6.0

Feature	Description
Application-Level Data Isolation	Each client app (Think Tank, Launch Board, AlwaysMe, etc.) is completely isolated
App-Scoped Users	Same email creates separate user instances per app - no cross-app data visibility
Enhanced RLS Policies	Row-Level Security now filters by BOTH `tenant_id` AND `app_id`
Separate Cognito User Pools	Each app gets its own authentication pool for complete identity isolation
App-Context Propagation	All API calls include `app_id` in context, enforced at Lambda level

Feature	Description
Think Tank Isolation	Think Tank is completely isolated from all other client apps
Cross-App Audit Trail	Administrators can view cross-app activity but users cannot
Migration Strategy	Non-breaking migration for existing deployments

Section 40: Application-Level Data Isolation

Why This Matters: - **Security**: A compromise in one app cannot affect another - **Compliance**: Per-app audit scope for HIPAA/SOC 2 - **Privacy**: Users in one app never see data from another app - **Data Sovereignty**: Clear boundaries for data residency requirements - **User Experience**: Clean, focused experience per application

Database Changes: - New `app_users` table linking users to specific apps - `app_id` column added to all user-facing data tables - Updated RLS policies with dual `tenant_id + app_id` filtering - New `current_app_id()` function for RLS context - Cross-app admin views for platform administrators

Infrastructure Changes: - Separate Cognito User Pool per application - App-specific JWT claims (`custom:app_id`) - Lambda context propagation for `app_id` - API Gateway routing per app subdomain

Lambda Changes: - Enhanced auth context extraction with `appId` - Database connection sets both `app.current_tenant_id` AND `app.current_app_id` - Audit logging includes `app_id` for all operations

---

Design Philosophy (v4.6.0)

- **Defense in Depth** - Isolation enforced at Cognito, API Gateway, Lambda, AND Database levels
- **Zero Trust** - Every request validates both tenant and app context
- **Least Privilege** - Users can only access their own app's data
- **Admin Override** - Platform admins can view across apps for support/debugging
- **Backward Compatible** - Existing single-app deployments work without changes

---

Also includes all v4.5.0 features:

WHAT'S NEW IN v4.5.0

Feature	Description
Dynamic Workflow Proposals	Brain & Neural Engine propose new workflows based on substantiated user needs
Evidence-Based Detection	7 evidence types: workflow_failure, negative_feedback, explicit_request, manual_override, repeated_attempt, support_escalation, pattern_detection

Feature	Description
Threshold-Gated Proposals	Min occurrences, unique users, time span, impact score, confidence thresholds
Brain Governor Review	All proposals pass through Brain risk assessment before admin queue
Admin Control	Human administrators approve all new workflows - no auto-publishing
Full Audit Trail	Complete tracking of evidence, decisions, and outcomes
Configurable Thresholds	Admin-editable occurrence, impact, and risk thresholds per tenant
4-Dimension Risk Assessment	Cost, latency, quality, compliance risks evaluated for each proposal

Also includes all v4.4.0 features:

Feature	Description
Neural-First Architecture	Neural Engine is the fabric, Brain is the governor - tight integration loop
Think Tank Workflow Registry	127 orchestration patterns, 127 production workflows, 834 specialized domains
Visual Workflow Editor	Comprehensive drag-and-drop orchestration builder with Neural connectors
Real-Time Steering	Neural Engine monitors and adjusts during execution, not just post-hoc
Per-User Neural Models	Personalized preference embeddings, domain preferences, behavioral patterns
Concurrent Execution Awareness	Full support for parallel user sessions in billing, feedback, and learning
Enhanced Analytics Integration	Analytics feeds Neural Engine learning signals in real-time
Full Admin Parameter Control	All Neural/Brain parameters editable through Admin Dashboard
Client Decision Transparency	Think Tank receives reasoning, confidence, alternatives for every decision
Swift Deployment App v2	Enhanced deployer with workflow management and Neural configuration

Also includes all v4.3.0 features:

Feature	Description
Feedback System	Thumbs up/down + optional categories + text/voice comments
Execution Manifests	Full provenance: models, orchestrations, services, thermal states, latency, cost

Feature	Description
Neural Engine Learning	Continuous learning from explicit + implicit feedback signals
Brain Neural Loop	Real-time Brain decisions informed by Neural Engine intelligence
Multi-Language Voice	Voice feedback in any language with auto-transcription/translation
Implicit Signal Capture	Regenerate, copy, abandon, manual switch = automatic feedback
Tiered Learning Scope	Individual → Tenant → Global learning with privacy isolation
Feedback Trust Scores	Anti-gaming: rate limits, outlier detection, weighted trust
A/B Testing Framework	Measure if routing changes actually improve outcomes
Cold Start Handling	Default routing + collaborative filtering for new users/models

Also includes all v4.2.0 features:

Feature	Description
Unified Model Registry	Single SQL view combining ALL 106 models (external + self-hosted)
Registry Sync Service	Automated Lambda syncs providers daily, health checks every 5 min
Complete Self-Hosted Catalog	56 self-hosted models with full metadata across 7 categories
Orchestration Selection	Smart model selection algorithm with thermal state awareness
hosting_type Field	Clear differentiation: 'external' vs 'self_hosted' per model
primary_mode Field	Routing mode: chat, completion, embedding, image, video, audio, search, 3d
Thermal-Aware Routing	Prefer HOT > WARM > COLD for latency optimization
Health Status Integration	Filter unhealthy providers/endpoints from selection

Also includes all v4.1.0 features:

Feature	Description
AlphaFold 2	Nobel Prize-winning protein folding (93M params, CASP14 champion)
Database-Driven Orchestration	ALL model configs stored in PostgreSQL - zero hardcoding
License Management	Track licenses, compliance status, and commercial use for all models

Feature	Description
Admin Model CRUD	Add/edit/delete models entirely through Admin Dashboard UI
Workflow Engine	Execute multi-step scientific workflows with database-defined DAGs
Audit Trail	Complete logging of all model, workflow, and license changes
Protein Folding Pipeline	Pre-built AlphaFold 2 workflow with MSA → Structure → Relaxation
Compliance Dashboard	Visual license compliance tracking with Apache/MIT/CC indicators

Also includes all v4.0.0 features:

Feature	Description
Time Machine	Apple Time Machine-inspired chat history - fly back through time
Visual Timeline	3D perspective view showing chat history fading into the past
Calendar Navigator	Jump to any date with visual calendar picker
Instant Restore	One-click restore of any message, file, or entire conversation state
Media Vault	Every file version preserved forever with S3 versioning
Service Layer APIs	Full Time Machine exposed via Complex API
AI Simplified API	Time Machine features available in simplified AI API
Never Lose Anything	Soft-delete only - everything recoverable forever
Export Bundles	Download complete history as ZIP/JSON/Markdown/PDF
Hidden by Default	Ultra-simple UI until user clicks "Enter Time Machine"

Also includes all v3.8.0 features:

Feature	Description
User Model Selection	Users can manually select AI models in Think Tank (not just Auto)
Standard Models (15)	Production-ready models from major providers
Novel Models (15)	Cutting-edge/experimental models with unique capabilities
Admin Editable Pricing	Full control over model pricing and markups in Admin Dashboard
Bulk Pricing Controls	Set all external (40% default) or self-hosted (75% default) to one margin
Individual Price Override	Override markup for specific models
Cost Transparency	Per-message cost display with user's selected model
Model Favorites	Users can favorite models for quick access

Feature	Description
Domain Mode Integration	Different default models per domain mode (Medical, Code, etc.)

VERSION HISTORY

Version	Sections	Key Features
v2.0.0	0-2	Foundation: Shared Types, Swift Deployer, CDK Infrastructure
v2.1.0	3-5	Core Platform: AI Stacks, Lambda Functions, Admin & Billing
v2.2.0	6-9	Full Platform: Self-Hosted Models, Database Schema, Admin Dashboard, Deployment
v2.3.0	10	Visual AI Pipeline (13 models)
v2.4.0	11	RADIANT Brain intelligent routing
v2.5.0	12	Metrics & Analytics Engine
v3.0.0	13	User Neural Engine (pgvector)
v3.1.0	14	Centralized Error Logging
v3.2.0	15-16	External & AWS Credentials Registry
v3.3.0	17	Simple Auto-Resolve API
v3.4.0	-	Internal milestone (no user-facing changes)
v3.5.0	18	THINK TANK consumer platform
v3.6.0	19-28	Concurrent Chat, Collaboration, Voice, Memory, Canvas, Personas, Scheduling, Family Plans, Analytics
v3.7.0	29-30	Admin Extensions, Dynamic Provider Registry + xAI/Grok
v3.8.0	31	Think Tank Model Selection & Editable Pricing
v4.0.0	32-33	Time Machine: Visual History, Service Layer APIs, Media Vault
v4.1.0	34-35	Database-Driven Orchestration, AlphaFold 2, License Management, Admin Model CRUD
v4.2.0	36	Unified Model Registry, Registry Sync Service, 56 Self-Hosted Models, Orchestration Selection
v4.3.0	37	Feedback Learning System, Neural Engine Loop, Multi-Language Voice, Implicit Signals, A/B Testing

Version	Sections	Key Features
v4.4.0	38	Neural-First Orchestration, Think Tank Workflow Registry, Visual Workflow Editor, Swift Deployer v2
v4.5.0	39	Dynamic Workflow Proposal System, Evidence-Based Detection, Brain Governor Review, Admin Control
v4.6.0	**40**	**Application-Level Data Isolation, App-Scoped Users, Enhanced RLS, Think Tank Isolation**
v4.7.0	**41**	**Complete i18n System, Localization Registry, AI Translation, 18 Languages**
v4.7.1	**-**	**Rename: Radiant Solver → Think Tank (consumer app branding)**
v4.8.0	**42**	**Dynamic Configuration Management, Admin-Editable Parameters, Hot Reload**
v4.9.0	**-**	**Structural Cleanup: Migration renumbering, duplicate removal, consistency fixes**
v4.10.0	**-**	**Implementation-Ready: Component clarification, phased implementation guide**
v4.12.0	**-**	**Brain Price Optimizer: Cost vs price distinction, caching/batch discounts**
v4.13.0	**43**	**Billing & Credits System - 7-tier subscriptions, prepaid credits, Stripe integration**
v4.14.0	**44**	**Storage Billing - Tiered S3/DB/backup pricing, quota management**
v4.15.0	**45**	**Versioned Subscriptions - Grandfathering, migration incentives**
v4.16.0	**46**	**Dual-Admin Approval - Two-person migration approval, audit trail**
v4.16.1	**-**	**Audit & Consistency Fixes - RLS standardization, type deduplication, localization**
v4.17.0	**-**	**AI Code Gen Optimization - Complete Swift app, Package.swift, Sendable conformance, version constants**

MIGRATION TO SECTION MAPPING

Migration	Section	Version	Description
001_initial_schema.sql	7	v2.2.0	Core tables: tenants, users, sessions
002_tenant_isolation.sql	7	v2.2.0	RLS policies for tenant isolation
003_ai_models.sql	7	v2.2.0	Models and providers tables
004_usage_billing.sql	7	v2.2.0	Usage events and billing
005_admin_approval.sql	5	v2.2.0	Admin invitations and approvals
006_self_hosted_models.sql	6	v2.2.0	SageMaker model configurations
007_external_providers.sql	7	v2.2.0	External provider registry
010_visual_ai_pipeline.sql	10	v2.3.0	Visual pipeline jobs
011_radiant_brain.sql	11	v2.4.0	Brain routing decisions
012_metrics_analytics.sql	12	v2.5.0	Usage metrics and aggregations
013_user_neural_engine.sql	13	v3.0.0	User preferences and memory
014_centralized_error_logging.sql	14	v3.1.0	Error logs table
015_credentials_registry.sql	15	v3.2.0	Credential vaults
016_aws_admin.sql	16	v3.2.0	AWS credentials integration
017_auto_resolve.sql	17	v3.3.0	Auto-resolve requests
018_think_tank.sql	18	v3.5.0	Think Tank sessions and steps
019_concurrent_chat.sql	19	v3.6.0	Concurrent sessions
020_realtime_collaboration.sql	20	v3.6.0	Collaboration sessions
021_voice_video.sql	21	v3.6.0	Voice sessions and transcriptions
022_persistent_memory.sql	22	v3.6.0	Memory stores
023_canvas_artifacts.sql	23	v3.6.0	Canvas and artifacts
024_result_merging.sql	24	v3.6.0	Merge sessions
025_focus_personas.sql	25	v3.6.0	User personas
026_scheduled_prompts.sql	26	v3.6.0	Scheduled prompts
027_team_plans.sql	27	v3.6.0	Team plans
028_analytics_extensions.sql	28	v3.6.0	Analytics dashboard extensions
029_admin_extensions.sql	29	v3.7.0	Admin dashboard extensions
030_dynamic_provider_registry.sql	30	v3.7.0	Provider registry with xAI
031_thinktank_model_selection.sql	31	v3.8.0	User model preferences
032_time_machine_core.sql	32	v4.0.0	Time Machine snapshots
033_time_machine_media.sql	33	v4.0.0	Media vault for Time Machine

Migration	Section	Version	Description
034_orchestration_engine.sql	34	v4.1.0	Orchestration patterns
035_license_management.sql	35	v4.1.0	License tracking
036_unified_model_registry.sql	36	v4.2.0	Unified model registry
037_feedback_learning.sql	37	v4.3.0	Feedback and learning signals
038_neural_orchestration.sql	38	v4.4.0	Neural orchestration registry
039_workflow_proposals.sql	39	v4.5.0	Dynamic workflow proposals
040_app_isolation.sql	40	v4.6.0	Application-level isolation
041_localization.sql	41	v4.7.0	Internationalization system
042_configuration.sql	42	v4.8.0	Dynamic configuration
043_billing_system.sql	43	v4.13.0	Billing tiers, credits, Stripe
044_storage_billing.sql	44	v4.14.0	Storage usage and pricing
045_versioned_subscriptions.sql	45	v4.15.0	Plan versions, grandfathering
046_dual_admin_approval.sql	46	v4.16.0	Migration approval workflow

---

IMPLEMENTATION DEPENDENCY GRAPH

Implementation Order (deploy in this sequence):

Phase 1: Foundation (Sections 0-2) - Section 0: Shared Types (no dependencies) - Section 1: Swift Deployment App (depends on: 0) - Section 2: CDK Infrastructure Stacks (depends on: 0)

Phase 2: Core Infrastructure (Sections 3-7) - Section 3: CDK AI & API Stacks (depends on: 2) - Section 4: Lambda Core (depends on: 0, 3) - Section 5: Lambda Admin & Billing (depends on: 4) - Section 6: Self-Hosted Models (depends on: 3) - Section 7: Database Schema (depends on: none - deploy migrations first!)

Phase 3: Admin & Deployment (Sections 8-9) - Section 8: Admin Dashboard (depends on: 4, 5, 7) - Section 9: Assembly & Deployment Guide (depends on: 1-8)

Phase 4: AI Features (Sections 10-17) - Section 10: Visual AI Pipeline (depends on: 7) - Section 11: RADIANT Brain (depends on: 10) - Section 12: Metrics & Analytics (depends on: 7) - Section 13: User Neural Engine (depends on: 7, 12) - Section 14: Error Logging (depends on: 7) - Section 15: Credentials Registry (depends on: 7) - Section 16: AWS Admin Credentials (depends on: 15) - Section 17: Auto-Resolve API (depends on: 11, 13)

Phase 5: Consumer Platform (Sections 18-28) - Section 18: Think Tank Platform (depends on: 11, 13) - Section 19: Concurrent Chat (depends on: 18) - Section 20: Real-Time Collaboration (depends on: 19) - Section 21: Voice & Video (depends on: 18) - Section 22: Persistent Memory (depends on: 13, 18) - Section 23: Canvas & Artifacts (depends on: 18) - Section 24: Result Merging (depends on: 19) - Section 25: Focus Modes & Personas (depends on: 18) - Section 26: Scheduled Prompts (depends on: 18) - Section 27: Family & Team Plans (depends on: 18) - Section 28: Analytics Integration (depends on: 12, 18)

Phase 6: Advanced Features (Sections 29-35) - Section 29: Admin Dashboard Extensions (depends on: 8, 18) - Section 30: Dynamic Provider Registry (depends on: 7, 11) - Section 31: Model Selection & Pricing (depends on: 18, 30) - Section 32: Time Machine Core (depends on: 18) - Section 33: Time Machine UI (depends on: 32) - Section 34: Orchestration Engine (depends on: 11, 30) - Section 35: License Management UI (depends on: 8, 34)

Phase 7: Intelligence Layer (Sections 36-39) - Section 36: Unified Model Registry (depends on: 30, 34) - Section 37: Feedback Learning (depends on: 13, 18) - Section 38: Neural Orchestration (depends on: 34, 36, 37) - Section 39: Workflow Proposals (depends on: 38)

Phase 8: Platform Hardening (Sections 40-42) - Section 40: App Isolation (depends on: all previous) - Section 41: Internationalization (depends on: 40) - Section 42: Dynamic Configuration (depends on: 41)

Phase 9: Billing & Enterprise (Sections 43-46) - Section 43: Billing & Credits (depends on: 42) - Section 44: Storage Billing (depends on: 43) - Section 45: Versioned Subscriptions (depends on: 43, 44) - Section 46: Dual-Admin Approval (depends on: 43)

CRITICAL: Database Connection Standard

ALL Lambda functions MUST use this pattern:

```
import { Pool } from 'pg';

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
  ssl: process.env.NODE_ENV === 'production' ? { rejectUnauthorized: false } : false,
});
```

Environment variable required:

```
DATABASE_URL=postgresql://user:password@host:5432/database?sslmode=require
```

Do NOT use individual DB_HOST, DB_NAME, DB_USER, DB_PASSWORD variables.

â• â