Contents

SECTION 8: ADMIN WEB DASHBOARD (v2.2.0) 2

â• â•

âš ï¸ IMPORTANT: Type Imports . . . 2

lib/api/types.ts . . . 2

RADIANT v2.2.0 - Prompt 8: Admin Web Dashboard (Next.js) 4

OVERVIEW . . . 4

ARCHITECTURE . . . 4

TECHNOLOGY STACK . . . 5

PROJECT STRUCTURE . . . 5

PART 1: PROJECT CONFIGURATION . . . 9

package.json . . . 9

next.config.js . . . 10

tailwind.config.ts . . . 12

app/globals.css . . . 14

.env.local.example . . . 18

PART 2: AUTHENTICATION . . . 19

lib/auth/amplify.ts . . . 19

lib/auth/context.tsx . . . 19

lib/auth/hooks.ts . . . 25

PART 3: API CLIENT . . . 26

lib/api/client.ts . . . 26

lib/api/types.ts . . . 29

lib/api/endpoints.ts . . . 36

PART 4: REACT QUERY HOOKS . . . 42

lib/hooks/use-models.ts . . . 42

lib/hooks/use-administrators.ts . . . 44

lib/hooks/use-billing.ts . . . 48

PART 5: LAYOUT COMPONENTS . . . 50

components/layout/sidebar.tsx . . . 50

components/layout/header.tsx . . . 55

components/layout/page-header.tsx . . . 56

PART 6: ROOT LAYOUT & PROVIDERS . . . 58

app/layout.tsx . . . 58

app/providers.tsx . . . 59

app/(dashboard)/layout.tsx . . . 60

PART 7: DASHBOARD PAGE . . . 61

app/(dashboard)/page.tsx . . . 61

app/(dashboard)/components/metrics-cards.tsx . . . 62

app/(dashboard)/components/system-health-card.tsx . . . 65

app/(dashboard)/components/quick-actions.tsx . . . 68

PART 8: MODELS PAGE . . . 69

app/(dashboard)/models/page.tsx . . . 69

app/(dashboard)/models/components/models-content.tsx . . . 70

components/models/model-card.tsx . . . 72

components/models/thermal-badge.tsx . . . 76
PART 9: ADMINISTRATORS & APPROVALS PAGES . . . 77
app/(dashboard)/administrators/page.tsx . . . 77
app/(dashboard)/administrators/approvals/page.tsx . . . 78
app/(dashboard)/administrators/approvals/components/approvals-content.tsx . . . 79
components/administrators/approval-card.tsx . . . 81
PART 10: UTILITY FUNCTIONS . . . 85
lib/utils/cn.ts . . . 85
lib/utils/format.ts . . . 85
lib/utils/constants.ts . . . 87
DEPLOYMENT . . . 87
Building for Production . . . 87
CDK Stack Integration . . . 88
NEXT PROMPTS . . . 88

â• â•

END OF SECTION 8 **88**

â• â•

â• â•

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;
```

```typescript
  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€ 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
Ã¢â€ â€š                                  ADMIN DASHBOARD LAYER                             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¢â€ â‚¬Ã¢â€ â
Ã¢â€ â€š                                                                                   Ã¢â€ â€š
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¢â€ â‚¬Ã¢
Ã¢â€ â€š   Ã¢â€ â€š                         NEXT.JS 14 APP                                Ã¢â€ â€
Ã¢â€ â€š   Ã¢â€ â€š   App Router Ã¢â€ â€š Server Components Ã¢â€ â€š Client Components
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
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¢â€ â‚¬Ã
Ã¢â€ â€š   Ã¢â€ â€š                  Ã¢â€ â€š                      Ã¢â€ â€š                     Ã¢â€ â€š
Ã¢â€ â€š   Ã¢â€"Â¼                  Ã¢â€"Â¼                      Ã¢â€"Â¼                     Ã¢â€"Â¼
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€ Å'Ã¢â€
Ã¢â€ â€š Ã¢â€ â€š  Auth   Ã¢â€ â€š  Ã¢â€ â€š  API   Ã¢â€ â€š      Ã¢â€ â€š  State    Ã¢â€ â€š    Ã
Ã¢â€ â€š Ã¢â€ â€šCognito Ã¢â€ â€š  Ã¢â€ â€š Client Ã¢â€ â€š      Ã¢â€ â€š React     Ã¢â€ â€š    Ã
Ã¢â€ â€š Ã¢â€ â€š  SDK   Ã¢â€ â€š  Ã¢â€ â€š  Fetch Ã¢â€ â€š      Ã¢â€ â€š Query     Ã¢â€ â€š    Ã
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€ Ëœ  Ã¢â€ â€ Ã¢â
Ã¢â€ â€š                                     Ã¢â€ â€š                                        Ã¢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€
                                          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¢â€ â
Ã¢â€ â€š                                      API GATEWAY                                  Ã¢â€ â€š
Ã¢â€ â€š                       REST API (/api/v2/*) Ã¢â€ â€š AppSync GraphQL                       Ã¢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€
```

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 reset
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 accep
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ layout.tsx               # Auth layout (centered,
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 with s
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 overv
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 list
```

```
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/edi
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 servic
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 provide
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 configu
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 list
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          # Pending i
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 appro
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 overview
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 co
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 history
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 analytics
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 set
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 setting
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 pattern
Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ globals.css                   # Design system tokens + Tailwi
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€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ progress.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ tooltip.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ switch.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ slider.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ alert.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ avatar.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ command.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ sheet.tsx
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ skeleton.tsx
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ layout/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ sidebar.tsx              # Main navigation sidebar
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ header.tsx               # Top header bar
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ page-header.tsx          # Page title + breadcrumb
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ mobile-nav.tsx           # Mobile navigation
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ dashboard/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ metric-card.tsx          # KPI display cards
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ activity-feed.tsx        # Recent activity
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ quick-actions.tsx        # Common actions
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ system-health.tsx        # Health overview
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ geographic/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ world-map.tsx            # Interactive world map
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ region-card.tsx           # Region statistics
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ latency-heatmap.tsx      # Latency visualization
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ models/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ model-card.tsx           # Model status card
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ thermal-badge.tsx        # Thermal state indicator
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ thermal-controls.tsx     # Thermal state managemen
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ model-filters.tsx        # Filter/search UI
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ model-table.tsx          # Models data table
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ pricing-form.tsx         # Price configuration
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ services/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ service-card.tsx          # Service status card
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ service-state-badge.tsx   # Service state indicator
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ dependency-tree.tsx       # Model dependencies
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ service-controls.tsx     # Service management
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ providers/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ provider-card.tsx         # Provider status
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ api-key-input.tsx        # Masked API key input
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ health-status.tsx        # Provider health check
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ administrators/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ admin-table.tsx          # Administrators list
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ invitation-form.tsx       # Send invitation modal
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ role-badge.tsx           # Role display
Ã¢â€šÃ¢â€šÃ¢â€â€Ã¢â€â¬Ã¢â€â¬ approval-card.tsx        # Approval request card
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ billing/
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ cost-chart.tsx           # Cost over time
Ã¢â€šÃ¢â€šÃ¢â€Å"Ã¢â€â¬Ã¢â€â¬ margin-slider.tsx        # Margin configuration
Ã¢â€šÃ¢â€šÃ¢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 icc
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ notification-list.tsx     # Notification dropdown
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 definition
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 configuration
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ Å"Ã¢â€ â‚¬Ã¢â€ â‚¬ context.tsx               # Auth context provider
Ã¢â€ â€š   Ã¢â€ â€š   Ã¢â€ â€ Ã¢â€ â‚¬Ã¢â€ â‚¬ hooks.ts                  # useAuth, useCurrentAdm
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 formatting
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://*.amaz
            ].join('; '),
          },
        ],
      },
    ];
  },
};

module.exports = nextConfig;
```

tailwind.config.ts

```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';
```

```tsx
import {
  createContext,
  useContext,
  useEffect,
  useState,
  useCallback,
  type ReactNode,
} from 'react';
import {
  getCurrentUser,
  fetchAuthSession,
  signIn,
  signOut,
  confirmSignIn,
  type AuthUser,
} from 'aws-amplify/auth';

// ============================================================================
// TYPES
// ============================================================================

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

```
/**
 * 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>): Promise
    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
// =============================================================================

export type ThermalState = 'OFF' | 'COLD' | 'WARM' | 'HOT' | 'AUTOMATIC';
```

```typescript
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_change
  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 | nu

  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, string

  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/providers/

  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<str

  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 }),
};
```

```typescript
// ============================================================================
// 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),
};

// ============================================================================
// 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

```
'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,
  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,
};

/**
 * 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';
```

```typescript
// 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
 */
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<MarginConfig>
      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/10">
            <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-mute
        >
          <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-col
          isActive
            ? 'bg-sidebar-accent text-sidebar-accent-foreground'
            : 'text-sidebar-foreground hover:bg-sidebar-accent hover:text-sidebar-accent-fore
        )}
      >
        <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-ac
                )}
              >
                <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-foregr
        )}
      >
        <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-primary
            {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-bac
      {/* 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-center
```

```
            <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:sc
              <Moon className="absolute h-4 w-4 rotate-90 scale-0 transition-all dark:rotate-0
              <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/10
              <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, ${metrics.activeModels.selfHosted}
      icon: Cpu,
      color: 'text-purple-500',
      bgColor: 'bg-purple-500/10',
    },
    {
      title: 'Revenue',
      value: formatCurrency(metrics.revenue.value),
      change: metrics.revenue.change,
      period: metrics.revenue.period,
      icon: DollarSign,
      color: 'text-green-500',
      bgColor: 'bg-green-500/10',
    },
    {
      title: 'Error Rate',
      value: `${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-foregr
          <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/10">
          <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-administrato
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•