Contents

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

â•

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

0.1 PROJECT ROOT CONFIGURATION

Directory Structure

```
radiant/
â"œâ"€â"€ package.json
â"œâ"€â"€ pnpm-workspace.yaml
â"œâ"€â"€ tsconfig.base.json
â"œâ"€â"€ .nvmrc
â"œâ"€â"€ .gitignore
â",
â"œâ"€â"€ packages/
â",   â"œâ"€â"€ shared/                    # SECTION 0 - Shared types (implement first)
â",   â""â"€â"€ infrastructure/            # SECTIONS 2-3 - CDK stacks
â",
â"œâ"€â"€ functions/                    # SECTIONS 4-5 - Lambda functions
â",
â"œâ"€â"€ migrations/                   # SECTION 7 - Database migrations
â",
â"œâ"€â"€ apps/
â",   â"œâ"€â"€ swift-deployer/           # SECTION 1 - macOS deployment app
â",   â""â"€â"€ admin-dashboard/          # SECTION 8 - Next.js admin UI
â",
â""â"€â"€ docs/
```

package.json (root)

```json
{
  "name": "radiant",
  "version": "2.2.0",
  "private": true,
  "workspaces": [
    "packages/*",
    "apps/*",
    "functions/*"
  ],
  "scripts": {
    "build": "pnpm -r build",
```

```
    "build:shared": "cd packages/shared && pnpm build",
    "test": "pnpm -r test",
    "lint": "pnpm -r lint",
    "deploy:dev": "cd packages/infrastructure && cdk deploy --all --context environment=dev",
    "deploy:staging": "cd packages/infrastructure && cdk deploy --all --context environment=sta
    "deploy:prod": "cd packages/infrastructure && cdk deploy --all --context environment=prod -
  },
  "devDependencies": {
    "@types/node": "^20.10.0",
    "typescript": "^5.3.0"
  },
  "engines": {
    "node": ">=20.0.0",
    "pnpm": ">=8.0.0"
  }
}
```

pnpm-workspace.yaml

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

tsconfig.base.json

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

.nvmrc

```
20
```

.gitignore

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

0.2 SHARED PACKAGE STRUCTURE

```
packages/shared/
â"œâ"€â"€ package.json
â"œâ"€â"€ tsconfig.json
â""â"€â"€ src/
    â"œâ"€â"€ index.ts
    â"œâ"€â"€ types/
    â"¦   â"œâ"€â"€ index.ts
    â"¦   â"œâ"€â"€ app.types.ts
    â"¦   â"œâ"€â"€ environment.types.ts
    â"¦   â"œâ"€â"€ ai.types.ts
    â"¦   â"œâ"€â"€ admin.types.ts
    â"¦   â"œâ"€â"€ billing.types.ts
    â"¦   â""â"€â"€ compliance.types.ts
    â"œâ"€â"€ constants/
    â"¦   â"œâ"€â"€ index.ts
    â"¦   â"œâ"€â"€ apps.ts
    â"¦   â"œâ"€â"€ environments.ts
    â"¦   â"œâ"€â"€ tiers.ts
    â"¦   â"œâ"€â"€ regions.ts
    â"¦   â""â"€â"€ providers.ts
    â""â"€â"€ utils/
        â"œâ"€â"€ index.ts
        â"œâ"€â"€ validation.ts
        â""â"€â"€ formatting.ts
```

packages/shared/package.json

```
{
  "name": "@radiant/shared",
  "version": "2.2.0",
```

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

packages/shared/tsconfig.json

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

0.3 VERSION & CONSTANTS

packages/shared/src/constants/version.ts

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

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

/** Current RADIANT platform version */
```

```typescript
export const RADIANT_VERSION = '4.17.0';

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

/** Minimum supported versions for various components */
export const MIN_VERSIONS = {
  node: '20.0.0',
  npm: '10.0.0',
  cdk: '2.120.0',
  postgres: '15.0',
  swift: '5.9',
  macos: '13.0',
  xcode: '15.0',
} as const;

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

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

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

packages/shared/src/constants/index.ts

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

0.4 TYPE DEFINITIONS

packages/shared/src/types/index.ts

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

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

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

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

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

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

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

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

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

export interface EnvironmentEndpoints {
  api?: string;
  graphql?: string;
  admin?: string;
  dashboard?: string;
}

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

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

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

export type DeploymentPhase =
  | 'idle'
  | 'initializing'
  | 'validating'
  | 'bootstrap'
  | 'foundation'
  | 'networking'
  | 'security'
```

```
  | 'data'
  | 'storage'
  | 'auth'
  | 'ai'
  | 'api'
  | 'admin'
  | 'migrations'
  | 'verification'
  | 'complete'
  | 'failed';

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

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

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

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

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

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

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

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

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

  // Compute
  vpcCidr: string;
  azCount: number;
  natGateways: number;

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

  // Cache
  elasticacheNodes: number;
  elasticacheNodeType: string;

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

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

  // Compliance
  enableHipaa: boolean;

  // Costs
  estimatedMonthlyCost: CostEstimate;
}

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

export interface RegionConfig {
  code: string;
  name: string;
```

```
  available: boolean;
  isGlobal: boolean;
}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

export interface Invitation {
  id: string;
  email: string;
  role: AdminRole;
  invitedBy: string;
  appId: string;
  tenantId: string;
  environment?: Environment;
```

```
  tokenHash: string;
  expiresAt: Date;
  status: InvitationStatus;
  message?: string;
  createdAt: Date;
  acceptedAt?: Date;
}

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

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

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

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

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

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

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

export interface UsageEvent {
```

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

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

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

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

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

export interface Invoice {
  id: string;
  tenantId: string;
  appId: string;
  stripeInvoiceId?: string;
  periodStart: Date;
  periodEnd: Date;
  subtotal: number;
  margin: number;
```

```
  tax: number;
  total: number;
  status: InvoiceStatus;
  paidAt?: Date;
  createdAt: Date;
}

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

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

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

export interface ProviderUsageSummary {
  providerId: string;
  providerName: string;
  requests: number;
  tokens: number;
  cost: number;
}

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

export interface TierDiscount {
```

```
  minTokens: number;
  discountPercent: number;
}

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

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

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

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

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

export type PHICategory =
  | 'name'
  | 'date'
  | 'phone'
  | 'email'
  | 'ssn'
  | 'mrn'
  | 'address'
  | 'age'
  | 'medical_condition'
  | 'medication'
  | 'procedure';

export const DEFAULT_PHI_CONFIG: PHIConfig = {
  mode: 'auto',
```

```
  categories: ['name', 'date', 'phone', 'email', 'ssn', 'mrn', 'address'],
  autoSanitize: true,
  allowReidentification: false,
  logSanitization: true,
  retentionDays: 365,
};

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

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

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

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

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

0.4 CONSTANTS

packages/shared/src/constants/index.ts

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

packages/shared/src/constants/apps.ts

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

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

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

packages/shared/src/constants/environments.ts

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

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

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

```
    minTier: 2 as TierLevel,
    defaultTier: 2 as TierLevel,
  },
  prod: {
    name: 'prod',
    displayName: 'Production',
    color: '#EF4444',
    requiresApproval: true,
    minTier: 3 as TierLevel,
    defaultTier: 3 as TierLevel,
  },
};

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

packages/shared/src/constants/tiers.ts

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

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

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

export const TIER_CONFIGS: Record<TierLevel, TierConfig> = {
  1: {
    level: 1,
    name: 'SEED',
    description: 'Development and testing, minimal costs',
    vpcCidr: '10.0.0.0/20',
    azCount: 2,
    natGateways: 1,
    auroraMinCapacity: 0.5,
    auroraMaxCapacity: 2,
    enableGlobalDatabase: false,
    elasticacheNodes: 0,
    elasticacheNodeType: 'cache.t4g.micro',
    enableSelfHostedModels: false,
    maxSagemakerEndpoints: 0,
    litellmTaskCount: 1,
```

```
  litellmCpu: 256,
  litellmMemory: 512,
  enableWaf: false,
  enableGuardDuty: false,
  enableSecurityHub: false,
  enableHipaa: false,
  estimatedMonthlyCost: { min: 50, max: 150, typical: 85 },
},
2: {
  level: 2,
  name: 'STARTUP',
  description: 'Small production workloads',
  vpcCidr: '10.0.0.0/18',
  azCount: 2,
  natGateways: 1,
  auroraMinCapacity: 1,
  auroraMaxCapacity: 8,
  enableGlobalDatabase: false,
  elasticacheNodes: 1,
  elasticacheNodeType: 'cache.t4g.small',
  enableSelfHostedModels: false,
  maxSagemakerEndpoints: 0,
  litellmTaskCount: 2,
  litellmCpu: 512,
  litellmMemory: 1024,
  enableWaf: true,
  enableGuardDuty: true,
  enableSecurityHub: false,
  enableHipaa: false,
  estimatedMonthlyCost: { min: 200, max: 400, typical: 255 },
},
3: {
  level: 3,
  name: 'GROWTH',
  description: 'Medium production with self-hosted models',
  vpcCidr: '10.0.0.0/17',
  azCount: 3,
  natGateways: 2,
  auroraMinCapacity: 2,
  auroraMaxCapacity: 16,
  enableGlobalDatabase: false,
  elasticacheNodes: 2,
  elasticacheNodeType: 'cache.r6g.large',
  enableSelfHostedModels: true,
  maxSagemakerEndpoints: 10,
  litellmTaskCount: 3,
  litellmCpu: 1024,
  litellmMemory: 2048,
```

```
  enableWaf: true,
  enableGuardDuty: true,
  enableSecurityHub: true,
  enableHipaa: true,
  estimatedMonthlyCost: { min: 1000, max: 2500, typical: 1475 },
},
4: {
  level: 4,
  name: 'SCALE',
  description: 'Large production with multi-region',
  vpcCidr: '10.0.0.0/16',
  azCount: 3,
  natGateways: 3,
  auroraMinCapacity: 4,
  auroraMaxCapacity: 64,
  enableGlobalDatabase: true,
  elasticacheNodes: 3,
  elasticacheNodeType: 'cache.r6g.xlarge',
  enableSelfHostedModels: true,
  maxSagemakerEndpoints: 30,
  litellmTaskCount: 5,
  litellmCpu: 2048,
  litellmMemory: 4096,
  enableWaf: true,
  enableGuardDuty: true,
  enableSecurityHub: true,
  enableHipaa: true,
  estimatedMonthlyCost: { min: 4000, max: 8000, typical: 5450 },
},
5: {
  level: 5,
  name: 'ENTERPRISE',
  description: 'Enterprise-grade global deployment',
  vpcCidr: '10.0.0.0/14',
  azCount: 3,
  natGateways: 3,
  auroraMinCapacity: 8,
  auroraMaxCapacity: 128,
  enableGlobalDatabase: true,
  elasticacheNodes: 6,
  elasticacheNodeType: 'cache.r6g.2xlarge',
  enableSelfHostedModels: true,
  maxSagemakerEndpoints: 100,
  litellmTaskCount: 10,
  litellmCpu: 4096,
  litellmMemory: 8192,
  enableWaf: true,
  enableGuardDuty: true,
```

```
    enableSecurityHub: true,
    enableHipaa: true,
    estimatedMonthlyCost: { min: 15000, max: 35000, typical: 21500 },
  },
};

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

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

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

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

packages/shared/src/constants/regions.ts

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

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

export const REGIONS: Record<string, RegionConfig> = {
  'us-east-1': { code: 'us-east-1', name: 'US East (N. Virginia)', available: true, isGlobal: t
```

```
  'us-west-2': { code: 'us-west-2', name: 'US West (Oregon)', available: true, isGlobal: false
  'eu-west-1': { code: 'eu-west-1', name: 'Europe (Ireland)', available: true, isGlobal: true
  'eu-central-1': { code: 'eu-central-1', name: 'Europe (Frankfurt)', available: true, isGlobal
  'ap-northeast-1': { code: 'ap-northeast-1', name: 'Asia Pacific (Tokyo)', available: true, is
  'ap-southeast-1': { code: 'ap-southeast-1', name: 'Asia Pacific (Singapore)', available: tru
  'ap-south-1': { code: 'ap-south-1', name: 'Asia Pacific (Mumbai)', available: true, isGlobal
};

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

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

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

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

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

packages/shared/src/constants/providers.ts

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

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

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

export const EXTERNAL_PROVIDERS: ExternalProviderInfo[] = [
  { id: 'openai', name: 'OpenAI', hipaaCompliant: true, capabilities: ['text_generation', 'chat
  { id: 'anthropic', name: 'Anthropic', hipaaCompliant: true, capabilities: ['text_generation'
  { id: 'google', name: 'Google AI', hipaaCompliant: true, capabilities: ['text_generation', '
  { id: 'xai', name: 'xAI', hipaaCompliant: false, capabilities: ['text_generation', 'chat_comp
  { id: 'perplexity', name: 'Perplexity', hipaaCompliant: false, capabilities: ['text_generatio
  { id: 'deepseek', name: 'DeepSeek', hipaaCompliant: false, capabilities: ['text_generation',
  { id: 'mistral', name: 'Mistral AI', hipaaCompliant: false, capabilities: ['text_generation'
  { id: 'cohere', name: 'Cohere', hipaaCompliant: true, capabilities: ['text_generation', 'chat
  { id: 'together', name: 'Together AI', hipaaCompliant: false, capabilities: ['text_generation
  { id: 'groq', name: 'Groq', hipaaCompliant: false, capabilities: ['text_generation', 'chat_co
  { id: 'fireworks', name: 'Fireworks AI', hipaaCompliant: false, capabilities: ['text_generati
  { id: 'replicate', name: 'Replicate', hipaaCompliant: false, capabilities: ['image_generation
  { id: 'huggingface', name: 'Hugging Face', hipaaCompliant: false, capabilities: ['text_genera
  { id: 'anyscale', name: 'Anyscale', hipaaCompliant: false, capabilities: ['text_generation',
  { id: 'databricks', name: 'Databricks', hipaaCompliant: true, capabilities: ['text_generation
  { id: 'aws_bedrock', name: 'AWS Bedrock', hipaaCompliant: true, capabilities: ['text_generati
  { id: 'azure_openai', name: 'Azure OpenAI', hipaaCompliant: true, capabilities: ['text_genera
  { id: 'vertex', name: 'Google Vertex AI', hipaaCompliant: true, capabilities: ['text_generati
  { id: 'ollama', name: 'Ollama (Local)', hipaaCompliant: true, capabilities: ['text_generation
  { id: 'elevenlabs', name: 'ElevenLabs', hipaaCompliant: false, capabilities: ['audio_generati
  { id: 'runway', name: 'Runway ML', hipaaCompliant: false, capabilities: ['video_generation',
];

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

```
  runway: 'https://api.runwayml.com/v1',
};

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

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

0.5 UTILITY FUNCTIONS

packages/shared/src/utils/index.ts

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

packages/shared/src/utils/validation.ts

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

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

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

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

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

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

export function sanitizeAppId(name: string): string {
  return name
    .toLowerCase()
```

```
    .replace(/[^a-z0-9-]/g, '-')
    .replace(/-+/g, '-')
    .replace(/^-|-$/g, '')
    .slice(0, 30);
}

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

packages/shared/src/utils/formatting.ts

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

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

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

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

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

  while (value >= 1024 && unitIndex < units.length - 1) {
```

```
    value /= 1024;
    unitIndex++;
  }

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

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

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

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

packages/shared/src/index.ts

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

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

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

// Utils
```

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

0.6 BUILD AND VERIFY

After implementing Section 0, build and verify:

```
cd packages/shared
pnpm install
pnpm build

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

Expected output structure:

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

â• â

END OF SECTION 0

â• â

â• â