Contents

SECTION 42: DYNAMIC CONFIGURATION MANAGEMENT SYSTEM (v4.8.0)

> **CRITICAL: This section eliminates ALL hardcoded runtime parameters. Every configurable value is now stored in the database and editable by admins.**

42.1 ARCHITECTURE OVERVIEW

The Problem

Before v4.8.0, RADIANT had numerous hardcoded parameters scattered throughout the codebase:

```
BEFORE (v4.7.x and earlier):

  Hardcoded Parameters Everywhere


   TypeScript Constants:
   const MAX_TOKENS = 128000;  ← Hardcoded!
   const DEFAULT_MARGIN = 0.40;  ← Hardcoded!
   const RETRY_ATTEMPTS = 3;  ← Hardcoded!

   Lambda Environment:
   timeout: Duration.seconds(30)  ← Hardcoded!
   memorySize: 512  ← Hardcoded!

   Rate Limits:
   maxConcurrent: 10  ← Hardcoded!
   maxPerMinute: 100  ← Hardcoded!

   Problems:
      Cannot adjust without code deployment
      No per-tenant customization
      No audit trail of changes
      Incident response requires developer
```

The Solution

v4.8.0 implements **complete database-driven configuration**:

```
AFTER (v4.8.0):

  Centralized Configuration Registry
```

```
system_configuration (Global Defaults)

  key: "pricing.external_provider_margin"
  value: 0.40
  category: "pricing"
  type: "decimal"
  min: 0.00, max: 1.00
```

```
tenant_configuration_overrides
   tenant_abc: 0.35 (negotiated discount)
   tenant_xyz: 0.50 (premium support)
   (others use global default 0.40)
```

```
Usage: getConfig('pricing.external_provider_margin', tenantId)
Returns: 0.35 for tenant_abc, 0.40 for others
```

Configuration Categories (12)

Category	Description	Example Parameters
`rate_limits`	API and service rate limits	requests_per_minute, concurrent_connections
`timeouts`	Request and processing timeouts	lambda_timeout, request_timeout, session_idle
`pricing`	Margins, markups, discounts	external_margin, self_hosted_margin, minimum_charge
`tokens`	Token and context limits	max_tokens_per_request, max_context_window
`retry`	Retry and backoff configuration	max_attempts, initial_delay, backoff_multiplier
`cache`	Cache TTL and invalidation	translation_bundle_ttl, model_list_ttl
`thresholds`	Health, confidence, quality thresholds	health_check_threshold, confidence_minimum
`discounts`	Volume discount tiers	tier_1_threshold, tier_1_discount
`session`	Auth and session settings	token_expiry, refresh_window, max_sessions
`translation`	i18n system settings	concurrent_limit, per_minute_limit

Category	Description	Example Parameters
`workflow`	Workflow proposal thresholds	min_occurrences, min_unique_users
`notifications`	Alert and notification settings	alert_thresholds, channels, cooldown

---

42.2 DATABASE SCHEMA

Migration: 042_configuration_management.sql

```sql
-- ============================================================================
-- RADIANT v4.8.0 - Dynamic Configuration Management System
-- Migration: 042_configuration_management.sql
-- ============================================================================

-- ============================================================================
-- 42.2.1 Configuration Categories
-- ============================================================================

CREATE TABLE configuration_categories (
    id VARCHAR(50) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    display_order INTEGER DEFAULT 100,
    icon VARCHAR(50),  -- Material UI icon name
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

INSERT INTO configuration_categories (id, name, description, display_order, icon) VALUES
('rate_limits', 'Rate Limits', 'API and service rate limiting configuration', 1, 'Speed'),
('timeouts', 'Timeouts', 'Request and processing timeout settings', 2, 'Timer'),
('pricing', 'Pricing', 'Margins, markups, and discount configuration', 3, 'AttachMoney'),
('tokens', 'Token Limits', 'Token and context window limits', 4, 'Token'),
('retry', 'Retry Configuration', 'Retry attempts and backoff settings', 5, 'Refresh'),
('cache', 'Cache Settings', 'Cache TTL and invalidation rules', 6, 'Cached'),
('thresholds', 'Thresholds', 'Health, confidence, and quality thresholds', 7, 'TrendingUp'),
('discounts', 'Volume Discounts', 'Volume-based discount tier configuration', 8, 'Discount'),
('session', 'Session & Auth', 'Authentication and session settings', 9, 'Lock'),
('translation', 'Translation System', 'i18n and translation service settings', 10, 'Translate'
('workflow', 'Workflow Proposals', 'Dynamic workflow proposal thresholds', 11, 'AccountTree'),
('notifications', 'Notifications', 'Alert thresholds and notification channels', 12, 'Notifica

-- ============================================================================
-- 42.2.2 Configuration Value Types
-- ============================================================================
```

```sql
CREATE TYPE config_value_type AS ENUM (
    'string',
    'integer',
    'decimal',
    'boolean',
    'json',
    'duration',      -- Stored as seconds, displayed as human-readable
    'percentage',    -- Stored as decimal (0.40 = 40%)
    'enum'
);

-- ============================================================================
-- 42.2.3 System Configuration (Global Defaults)
-- ============================================================================

CREATE TABLE system_configuration (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Identification
    key VARCHAR(100) NOT NULL UNIQUE,
    category_id VARCHAR(50) NOT NULL REFERENCES configuration_categories(id),

    -- Value storage
    value_type config_value_type NOT NULL,
    value_string TEXT,            -- For string, enum types
    value_integer BIGINT,         -- For integer, duration types
    value_decimal DECIMAL(20,6), -- For decimal, percentage types
    value_boolean BOOLEAN,        -- For boolean type
    value_json JSONB,             -- For json type

    -- Metadata
    display_name VARCHAR(200) NOT NULL,
    description TEXT,
    unit VARCHAR(50),             -- e.g., 'seconds', 'requests', '%', 'tokens'

    -- Validation constraints
    min_value DECIMAL(20,6),
    max_value DECIMAL(20,6),
    enum_values TEXT[],           -- For enum type
    regex_pattern TEXT,           -- For string validation

    -- Environment scoping
    environment VARCHAR(20) DEFAULT 'all',  -- 'all', 'dev', 'staging', 'prod'

    -- Feature flags
    is_sensitive BOOLEAN DEFAULT FALSE,      -- Mask value in UI
    requires_restart BOOLEAN DEFAULT FALSE, -- Warn admin
    is_deprecated BOOLEAN DEFAULT FALSE,
```

```
    deprecated_replacement_key VARCHAR(100),

    -- Audit
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_by UUID REFERENCES administrators(id),

    -- Constraints
    CONSTRAINT valid_value CHECK (
        (value_type = 'string' AND value_string IS NOT NULL) OR
        (value_type = 'integer' AND value_integer IS NOT NULL) OR
        (value_type = 'decimal' AND value_decimal IS NOT NULL) OR
        (value_type = 'boolean' AND value_boolean IS NOT NULL) OR
        (value_type = 'json' AND value_json IS NOT NULL) OR
        (value_type = 'duration' AND value_integer IS NOT NULL) OR
        (value_type = 'percentage' AND value_decimal IS NOT NULL) OR
        (value_type = 'enum' AND value_string IS NOT NULL)
    )
);

CREATE INDEX idx_system_config_category ON system_configuration(category_id);
CREATE INDEX idx_system_config_key ON system_configuration(key);
CREATE INDEX idx_system_config_env ON system_configuration(environment);

-- ============================================================================
-- 42.2.4 Tenant Configuration Overrides
-- ============================================================================

CREATE TABLE tenant_configuration_overrides (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    config_id UUID NOT NULL REFERENCES system_configuration(id) ON DELETE CASCADE,

    -- Override value (same structure as system_configuration)
    value_string TEXT,
    value_integer BIGINT,
    value_decimal DECIMAL(20,6),
    value_boolean BOOLEAN,
    value_json JSONB,

    -- Validity period (optional)
    valid_from TIMESTAMPTZ DEFAULT NOW(),
    valid_until TIMESTAMPTZ,  -- NULL = forever

    -- Audit
    reason TEXT,              -- Why this override exists
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
```

```
    created_by UUID REFERENCES administrators(id),
    updated_by UUID REFERENCES administrators(id),

    UNIQUE(tenant_id, config_id)
);

CREATE INDEX idx_tenant_config_tenant ON tenant_configuration_overrides(tenant_id);
CREATE INDEX idx_tenant_config_config ON tenant_configuration_overrides(config_id);
CREATE INDEX idx_tenant_config_validity ON tenant_configuration_overrides(valid_from, valid_un

-- RLS
ALTER TABLE tenant_configuration_overrides ENABLE ROW LEVEL SECURITY;

CREATE POLICY tenant_config_isolation ON tenant_configuration_overrides
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

-- ============================================================================
-- 42.2.5 Configuration Audit Log
-- ============================================================================

CREATE TABLE configuration_audit_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- What changed
    config_id UUID REFERENCES system_configuration(id) ON DELETE SET NULL,
    tenant_override_id UUID REFERENCES tenant_configuration_overrides(id) ON DELETE SET NULL,
    config_key VARCHAR(100) NOT NULL,
    tenant_id UUID,  -- NULL for global changes

    -- Change details
    action VARCHAR(50) NOT NULL,  -- 'created', 'updated', 'deleted', 'override_created', 'ove
    old_value JSONB,
    new_value JSONB,

    -- Who made the change
    changed_by UUID REFERENCES administrators(id),
    changed_by_email VARCHAR(255),

    -- Context
    reason TEXT,
    ip_address INET,
    user_agent TEXT,

    -- Timestamps
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_config_audit_config ON configuration_audit_log(config_id);
```

```sql
CREATE INDEX idx_config_audit_tenant ON configuration_audit_log(tenant_id);
CREATE INDEX idx_config_audit_created ON configuration_audit_log(created_at DESC);
CREATE INDEX idx_config_audit_key ON configuration_audit_log(config_key);

-- ============================================================================
-- 42.2.6 Configuration Cache Invalidation
-- ============================================================================

CREATE TABLE configuration_cache_invalidation (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    config_key VARCHAR(100) NOT NULL,
    tenant_id UUID,  -- NULL = global invalidation
    invalidated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    processed_at TIMESTAMPTZ,
    processed_by VARCHAR(100)  -- Lambda function name that processed it
);

CREATE INDEX idx_config_cache_pending ON configuration_cache_invalidation(invalidated_at)
    WHERE processed_at IS NULL;

-- ============================================================================
-- 42.2.7 Helper Functions
-- ============================================================================

-- Get configuration value with tenant override support
CREATE OR REPLACE FUNCTION get_config(
    p_key VARCHAR(100),
    p_tenant_id UUID DEFAULT NULL,
    p_environment VARCHAR(20) DEFAULT 'prod'
) RETURNS JSONB AS $$
DECLARE
    v_config system_configuration%ROWTYPE;
    v_override tenant_configuration_overrides%ROWTYPE;
    v_result JSONB;
BEGIN
    -- Get base configuration
    SELECT * INTO v_config
    FROM system_configuration
    WHERE key = p_key
      AND (environment = 'all' OR environment = p_environment);

    IF v_config IS NULL THEN
        RETURN NULL;
    END IF;

    -- Build base result
    v_result = jsonb_build_object(
        'key', v_config.key,
```

```
        'type', v_config.value_type,
        'value', CASE v_config.value_type
            WHEN 'string' THEN to_jsonb(v_config.value_string)
            WHEN 'integer' THEN to_jsonb(v_config.value_integer)
            WHEN 'decimal' THEN to_jsonb(v_config.value_decimal)
            WHEN 'boolean' THEN to_jsonb(v_config.value_boolean)
            WHEN 'json' THEN v_config.value_json
            WHEN 'duration' THEN to_jsonb(v_config.value_integer)
            WHEN 'percentage' THEN to_jsonb(v_config.value_decimal)
            WHEN 'enum' THEN to_jsonb(v_config.value_string)
        END,
        'is_override', false
    );

    -- Check for tenant override if tenant_id provided
    IF p_tenant_id IS NOT NULL THEN
        SELECT * INTO v_override
        FROM tenant_configuration_overrides
        WHERE config_id = v_config.id
          AND tenant_id = p_tenant_id
          AND valid_from <= NOW()
          AND (valid_until IS NULL OR valid_until > NOW());

        IF v_override IS NOT NULL THEN
            v_result = jsonb_set(v_result, '{value}',
                CASE v_config.value_type
                    WHEN 'string' THEN to_jsonb(v_override.value_string)
                    WHEN 'integer' THEN to_jsonb(v_override.value_integer)
                    WHEN 'decimal' THEN to_jsonb(v_override.value_decimal)
                    WHEN 'boolean' THEN to_jsonb(v_override.value_boolean)
                    WHEN 'json' THEN v_override.value_json
                    WHEN 'duration' THEN to_jsonb(v_override.value_integer)
                    WHEN 'percentage' THEN to_jsonb(v_override.value_decimal)
                    WHEN 'enum' THEN to_jsonb(v_override.value_string)
                END
            );
            v_result = jsonb_set(v_result, '{is_override}', 'true'::jsonb);
        END IF;
    END IF;

    RETURN v_result;
END;
$$ LANGUAGE plpgsql STABLE;

-- Get all configurations for a category
CREATE OR REPLACE FUNCTION get_configs_by_category(
    p_category VARCHAR(50),
    p_tenant_id UUID DEFAULT NULL,
```

```
    p_environment VARCHAR(20) DEFAULT 'prod'
) RETURNS TABLE (
    key VARCHAR(100),
    value JSONB,
    display_name VARCHAR(200),
    description TEXT,
    value_type config_value_type,
    unit VARCHAR(50),
    is_override BOOLEAN
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        sc.key,
        (get_config(sc.key, p_tenant_id, p_environment))->>'value',
        sc.display_name,
        sc.description,
        sc.value_type,
        sc.unit,
        ((get_config(sc.key, p_tenant_id, p_environment))->>'is_override')::BOOLEAN
    FROM system_configuration sc
    WHERE sc.category_id = p_category
      AND (sc.environment = 'all' OR sc.environment = p_environment)
      AND sc.is_deprecated = false
    ORDER BY sc.key;
END;
$$ LANGUAGE plpgsql STABLE;

-- Trigger to log configuration changes
CREATE OR REPLACE FUNCTION log_config_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO configuration_audit_log (
            config_id, config_key, action, new_value, changed_by
        ) VALUES (
            NEW.id, NEW.key, 'created',
            jsonb_build_object('value', COALESCE(
                to_jsonb(NEW.value_string),
                to_jsonb(NEW.value_integer),
                to_jsonb(NEW.value_decimal),
                to_jsonb(NEW.value_boolean),
                NEW.value_json
            )),
            NEW.updated_by
        );
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO configuration_audit_log (
```

```
            config_id, config_key, action, old_value, new_value, changed_by
        ) VALUES (
            NEW.id, NEW.key, 'updated',
            jsonb_build_object('value', COALESCE(
                to_jsonb(OLD.value_string),
                to_jsonb(OLD.value_integer),
                to_jsonb(OLD.value_decimal),
                to_jsonb(OLD.value_boolean),
                OLD.value_json
            )),
            jsonb_build_object('value', COALESCE(
                to_jsonb(NEW.value_string),
                to_jsonb(NEW.value_integer),
                to_jsonb(NEW.value_decimal),
                to_jsonb(NEW.value_boolean),
                NEW.value_json
            )),
            NEW.updated_by
        );

        -- Queue cache invalidation
        INSERT INTO configuration_cache_invalidation (config_key)
        VALUES (NEW.key);
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_config_changes
    AFTER INSERT OR UPDATE ON system_configuration
    FOR EACH ROW
    EXECUTE FUNCTION log_config_changes();

-- Trigger for tenant override changes
CREATE OR REPLACE FUNCTION log_tenant_override_changes()
RETURNS TRIGGER AS $$
DECLARE
    v_config_key VARCHAR(100);
BEGIN
    SELECT key INTO v_config_key FROM system_configuration WHERE id = COALESCE(NEW.config_id, (

    IF TG_OP = 'INSERT' THEN
        INSERT INTO configuration_audit_log (
            tenant_override_id, config_key, tenant_id, action, new_value, changed_by, reason
        ) VALUES (
            NEW.id, v_config_key, NEW.tenant_id, 'override_created',
            jsonb_build_object('value', COALESCE(
```

```
            to_jsonb(NEW.value_string),
            to_jsonb(NEW.value_integer),
            to_jsonb(NEW.value_decimal),
            to_jsonb(NEW.value_boolean),
            NEW.value_json
        )),
        NEW.created_by, NEW.reason
    );
ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO configuration_audit_log (
        tenant_override_id, config_key, tenant_id, action, old_value, new_value, changed_by
    ) VALUES (
        NEW.id, v_config_key, NEW.tenant_id, 'override_updated',
        jsonb_build_object('value', COALESCE(
            to_jsonb(OLD.value_string),
            to_jsonb(OLD.value_integer),
            to_jsonb(OLD.value_decimal),
            to_jsonb(OLD.value_boolean),
            OLD.value_json
        )),
        jsonb_build_object('value', COALESCE(
            to_jsonb(NEW.value_string),
            to_jsonb(NEW.value_integer),
            to_jsonb(NEW.value_decimal),
            to_jsonb(NEW.value_boolean),
            NEW.value_json
        )),
        NEW.updated_by, NEW.reason
    );
ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO configuration_audit_log (
        config_key, tenant_id, action, old_value, changed_by
    ) VALUES (
        v_config_key, OLD.tenant_id, 'override_deleted',
        jsonb_build_object('value', COALESCE(
            to_jsonb(OLD.value_string),
            to_jsonb(OLD.value_integer),
            to_jsonb(OLD.value_decimal),
            to_jsonb(OLD.value_boolean),
            OLD.value_json
        )),
        current_setting('app.current_admin_id', true)::UUID
    );
END IF;

-- Queue cache invalidation
INSERT INTO configuration_cache_invalidation (config_key, tenant_id)
VALUES (v_config_key, COALESCE(NEW.tenant_id, OLD.tenant_id));
```

```
    RETURN COALESCE(NEW, OLD);
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_tenant_override_changes
    AFTER INSERT OR UPDATE OR DELETE ON tenant_configuration_overrides
    FOR EACH ROW
    EXECUTE FUNCTION log_tenant_override_changes();
```

42.3 SEED CONFIGURATION DATA

Migration: 042b_seed_configuration.sql

```
-- ============================================================================
-- RADIANT v4.8.0 - Seed Initial Configuration Values
-- Migration: 042b_seed_configuration.sql
-- ============================================================================

-- ============================================================================
-- Rate Limits
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('rate_limits.api.requests_per_minute', 'rate_limits', 'integer', 1000, 'API Requests per Minut
('rate_limits.api.requests_per_hour', 'rate_limits', 'integer', 50000, 'API Requests per Hour'
('rate_limits.api.concurrent_connections', 'rate_limits', 'integer', 100, 'Concurrent Connectio
('rate_limits.chat.messages_per_minute', 'rate_limits', 'integer', 60, 'Chat Messages per Minut
('rate_limits.chat.concurrent_sessions', 'rate_limits', 'integer', 5, 'Concurrent Chat Sessions
('rate_limits.file_upload.max_size_mb', 'rate_limits', 'integer', 100, 'Max File Upload Size',
('rate_limits.file_upload.per_hour', 'rate_limits', 'integer', 50, 'File Uploads per Hour', 'Ma

-- ============================================================================
-- Timeouts
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('timeouts.lambda.default', 'timeouts', 'duration', 30, 'Default Lambda Timeout', 'Default time
('timeouts.lambda.chat', 'timeouts', 'duration', 120, 'Chat Lambda Timeout', 'Timeout for chat
('timeouts.lambda.orchestration', 'timeouts', 'duration', 300, 'Orchestration Lambda Timeout',
('timeouts.lambda.batch', 'timeouts', 'duration', 900, 'Batch Processing Timeout', 'Timeout for
('timeouts.request.api_gateway', 'timeouts', 'duration', 29, 'API Gateway Timeout', 'API Gatewa
('timeouts.request.external_provider', 'timeouts', 'duration', 60, 'External Provider Timeout'
('timeouts.session.idle', 'timeouts', 'duration', 1800, 'Session Idle Timeout', 'Time before id
('timeouts.session.absolute', 'timeouts', 'duration', 86400, 'Absolute Session Timeout', 'Maxim

-- ============================================================================
```

```
-- Pricing
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name, de
('pricing.external_provider_margin', 'pricing', 'percentage', 0.40, 'External Provider Margin'
('pricing.self_hosted_margin', 'pricing', 'percentage', 0.75, 'Self-Hosted Margin', 'Default ma
('pricing.minimum_charge', 'pricing', 'decimal', 0.01, 'Minimum Charge', 'Minimum charge per tr
('pricing.tax_rate_default', 'pricing', 'percentage', 0.00, 'Default Tax Rate', 'Default tax ra

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('pricing.invoice.due_days', 'pricing', 'integer', 30, 'Invoice Due Days', 'Days until invoice
('pricing.invoice.reminder_days', 'pricing', 'integer', 7, 'Invoice Reminder Days', 'Days befor

-- ============================================================================
-- Token Limits
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('tokens.max_per_request', 'tokens', 'integer', 128000, 'Max Tokens per Request', 'Maximum toke
('tokens.max_context_window', 'tokens', 'integer', 200000, 'Max Context Window', 'Maximum conte
('tokens.max_output', 'tokens', 'integer', 8192, 'Max Output Tokens', 'Maximum tokens in respon
('tokens.streaming_chunk_size', 'tokens', 'integer', 100, 'Streaming Chunk Size', 'Tokens per s

-- ============================================================================
-- Retry Configuration
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('retry.max_attempts', 'retry', 'integer', 3, 'Max Retry Attempts', 'Maximum number of retry at
('retry.initial_delay_ms', 'retry', 'integer', 1000, 'Initial Retry Delay', 'Initial delay befo
('retry.max_delay_ms', 'retry', 'integer', 30000, 'Max Retry Delay', 'Maximum delay between ret

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name, de
('retry.backoff_multiplier', 'retry', 'decimal', 2.0, 'Backoff Multiplier', 'Multiplier for exp
('retry.jitter_factor', 'retry', 'decimal', 0.1, 'Jitter Factor', 'Random jitter added to delay

-- ============================================================================
-- Cache Settings
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('cache.translation_bundle_ttl', 'cache', 'duration', 300, 'Translation Bundle TTL', 'Cache dur
('cache.model_list_ttl', 'cache', 'duration', 300, 'Model List TTL', 'Cache duration for AI mod
('cache.provider_health_ttl', 'cache', 'duration', 60, 'Provider Health TTL', 'Cache duration f
('cache.user_preferences_ttl', 'cache', 'duration', 600, 'User Preferences TTL', 'Cache duratio
('cache.config_ttl', 'cache', 'duration', 300, 'Configuration TTL', 'Cache duration for system

-- ============================================================================
```

```
-- Thresholds
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('thresholds.health_check.healthy', 'thresholds', 'integer', 2, 'Healthy Threshold', 'Consecuti
('thresholds.health_check.unhealthy', 'thresholds', 'integer', 3, 'Unhealthy Threshold', 'Conse

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name, de
('thresholds.confidence.minimum', 'thresholds', 'percentage', 0.70, 'Minimum Confidence', 'Mini
('thresholds.confidence.high', 'thresholds', 'percentage', 0.90, 'High Confidence', 'Threshold
('thresholds.autoscaling.target_utilization', 'thresholds', 'percentage', 0.70, 'Target Utiliza

-- ============================================================================
-- Volume Discounts
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('discounts.tier_1.threshold', 'discounts', 'integer', 1000000, 'Tier 1 Threshold', 'Token thre
('discounts.tier_2.threshold', 'discounts', 'integer', 10000000, 'Tier 2 Threshold', 'Token thr
('discounts.tier_3.threshold', 'discounts', 'integer', 100000000, 'Tier 3 Threshold', 'Token th

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name, de
('discounts.tier_1.percentage', 'discounts', 'percentage', 0.05, 'Tier 1 Discount', 'Discount p
('discounts.tier_2.percentage', 'discounts', 'percentage', 0.10, 'Tier 2 Discount', 'Discount p
('discounts.tier_3.percentage', 'discounts', 'percentage', 0.15, 'Tier 3 Discount', 'Discount p

-- ============================================================================
-- Session & Auth
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('session.token_expiry', 'session', 'duration', 3600, 'Token Expiry', 'Access token expiration
('session.refresh_token_expiry', 'session', 'duration', 2592000, 'Refresh Token Expiry', 'Refre
('session.max_per_user', 'session', 'integer', 5, 'Max Sessions per User', 'Maximum concurrent
('session.invitation_expiry', 'session', 'duration', 604800, 'Invitation Expiry', 'Admin invita

-- ============================================================================
-- Translation System
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('translation.concurrent_limit', 'translation', 'integer', 10, 'Concurrent Translations', 'Maxi
('translation.per_minute_limit', 'translation', 'integer', 100, 'Translations per Minute', 'Max
('translation.per_hour_limit', 'translation', 'integer', 1000, 'Translations per Hour', 'Maximu
('translation.retry_attempts', 'translation', 'integer', 3, 'Translation Retry Attempts', 'Numb
('translation.retry_delay_ms', 'translation', 'integer', 1000, 'Translation Retry Delay', 'Init
('translation.queue_batch_size', 'translation', 'integer', 50, 'Translation Queue Batch', 'Numb
```

```sql
-- ============================================================================
-- Workflow Proposals
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('workflow.proposal.min_occurrences', 'workflow', 'integer', 5, 'Min Occurrences', 'Minimum pat
('workflow.proposal.min_unique_users', 'workflow', 'integer', 3, 'Min Unique Users', 'Minimum u
('workflow.proposal.min_time_span_hours', 'workflow', 'integer', 24, 'Min Time Span', 'Minimum
('workflow.proposal.max_per_day', 'workflow', 'integer', 10, 'Max Proposals per Day', 'Maximum
('workflow.proposal.max_per_week', 'workflow', 'integer', 30, 'Max Proposals per Week', 'Maximu

INSERT INTO system_configuration (key, category_id, value_type, value_decimal, display_name, de
('workflow.proposal.min_impact_score', 'workflow', 'percentage', 0.60, 'Min Impact Score', 'Mi
('workflow.proposal.min_confidence', 'workflow', 'percentage', 0.75, 'Min Confidence', 'Minimum

-- ============================================================================
-- Notifications
-- ============================================================================

INSERT INTO system_configuration (key, category_id, value_type, value_json, display_name, descr
('notifications.usage_alert_thresholds', 'notifications', 'json', '[50, 75, 90, 100]', 'Usage A

INSERT INTO system_configuration (key, category_id, value_type, value_integer, display_name, de
('notifications.alert_cooldown', 'notifications', 'duration', 3600, 'Alert Cooldown', 'Minimum
('notifications.digest_frequency', 'notifications', 'duration', 86400, 'Digest Frequency', 'Fre

INSERT INTO system_configuration (key, category_id, value_type, value_boolean, display_name, de
('notifications.email_enabled', 'notifications', 'boolean', true, 'Email Notifications', 'Enabl
('notifications.slack_enabled', 'notifications', 'boolean', false, 'Slack Notifications', 'Enab
('notifications.webhook_enabled', 'notifications', 'boolean', false, 'Webhook Notifications',
```

42.4 TYPESCRIPT TYPES

File: packages/shared/src/config/types.ts

```typescript
// ============================================================================
// RADIANT v4.8.0 - Configuration Management Types
// ============================================================================

/**
 * Configuration value types
 */
export type ConfigValueType =
  | 'string'
  | 'integer'
  | 'decimal'
  | 'boolean'
```

```
  | 'json'
  | 'duration'
  | 'percentage'
  | 'enum';

/**
 * Configuration category
 */
export interface ConfigCategory {
  id: string;
  name: string;
  description?: string;
  displayOrder: number;
  icon?: string;
}

/**
 * System configuration entry
 */
export interface SystemConfig {
  id: string;
  key: string;
  categoryId: string;
  valueType: ConfigValueType;
  value: string | number | boolean | object;
  displayName: string;
  description?: string;
  unit?: string;
  minValue?: number;
  maxValue?: number;
  enumValues?: string[];
  regexPattern?: string;
  environment: 'all' | 'dev' | 'staging' | 'prod';
  isSensitive: boolean;
  requiresRestart: boolean;
  isDeprecated: boolean;
  deprecatedReplacementKey?: string;
  createdAt: string;
  updatedAt: string;
  updatedBy?: string;
}

/**
 * Tenant configuration override
 */
export interface TenantConfigOverride {
  id: string;
  tenantId: string;
```

```typescript
  configId: string;
  value: string | number | boolean | object;
  validFrom: string;
  validUntil?: string;
  reason?: string;
  createdAt: string;
  updatedAt: string;
  createdBy?: string;
  updatedBy?: string;
}

/**
 * Configuration with resolved value (after tenant override)
 */
export interface ResolvedConfig {
  key: string;
  value: string | number | boolean | object;
  type: ConfigValueType;
  isOverride: boolean;
  displayName?: string;
  unit?: string;
}

/**
 * Configuration audit log entry
 */
export interface ConfigAuditEntry {
  id: string;
  configId?: string;
  tenantOverrideId?: string;
  configKey: string;
  tenantId?: string;
  action: 'created' | 'updated' | 'deleted' | 'override_created' | 'override_updated' | 'overri
  oldValue?: object;
  newValue?: object;
  changedBy?: string;
  changedByEmail?: string;
  reason?: string;
  ipAddress?: string;
  userAgent?: string;
  createdAt: string;
}

/**
 * Configuration update request
 */
export interface ConfigUpdateRequest {
  value: string | number | boolean | object;
```

```
  reason?: string;
}

/**
 * Tenant override request
 */
export interface TenantOverrideRequest {
  value: string | number | boolean | object;
  reason?: string;
  validFrom?: string;
  validUntil?: string;
}

/**
 * Configuration export/import format
 */
export interface ConfigExport {
  version: string;
  exportedAt: string;
  exportedBy: string;
  configs: Array<{
    key: string;
    value: string | number | boolean | object;
  }>;
  tenantOverrides?: Array<{
    tenantId: string;
    key: string;
    value: string | number | boolean | object;
    reason?: string;
  }>;
}
```

File: packages/shared/src/config/constants.ts

```
// =============================================================================
// RADIANT v4.8.0 - Configuration Constants
// =============================================================================

/**
 * Configuration categories
 */
export const CONFIG_CATEGORIES = {
  RATE_LIMITS: 'rate_limits',
  TIMEOUTS: 'timeouts',
  PRICING: 'pricing',
  TOKENS: 'tokens',
  RETRY: 'retry',
  CACHE: 'cache',
```

```
  THRESHOLDS: 'thresholds',
  DISCOUNTS: 'discounts',
  SESSION: 'session',
  TRANSLATION: 'translation',
  WORKFLOW: 'workflow',
  NOTIFICATIONS: 'notifications',
} as const;

/**
 * Common configuration keys
 */
export const CONFIG_KEYS = {
  // Rate Limits
  API_REQUESTS_PER_MINUTE: 'rate_limits.api.requests_per_minute',
  API_REQUESTS_PER_HOUR: 'rate_limits.api.requests_per_hour',
  API_CONCURRENT_CONNECTIONS: 'rate_limits.api.concurrent_connections',
  CHAT_MESSAGES_PER_MINUTE: 'rate_limits.chat.messages_per_minute',

  // Timeouts
  LAMBDA_DEFAULT_TIMEOUT: 'timeouts.lambda.default',
  LAMBDA_CHAT_TIMEOUT: 'timeouts.lambda.chat',
  EXTERNAL_PROVIDER_TIMEOUT: 'timeouts.request.external_provider',
  SESSION_IDLE_TIMEOUT: 'timeouts.session.idle',

  // Pricing
  EXTERNAL_PROVIDER_MARGIN: 'pricing.external_provider_margin',
  SELF_HOSTED_MARGIN: 'pricing.self_hosted_margin',
  MINIMUM_CHARGE: 'pricing.minimum_charge',

  // Tokens
  MAX_TOKENS_PER_REQUEST: 'tokens.max_per_request',
  MAX_CONTEXT_WINDOW: 'tokens.max_context_window',
  MAX_OUTPUT_TOKENS: 'tokens.max_output',

  // Retry
  MAX_RETRY_ATTEMPTS: 'retry.max_attempts',
  INITIAL_RETRY_DELAY: 'retry.initial_delay_ms',
  BACKOFF_MULTIPLIER: 'retry.backoff_multiplier',

  // Cache
  TRANSLATION_BUNDLE_TTL: 'cache.translation_bundle_ttl',
  MODEL_LIST_TTL: 'cache.model_list_ttl',
  CONFIG_TTL: 'cache.config_ttl',

  // Thresholds
  MIN_CONFIDENCE: 'thresholds.confidence.minimum',
  HIGH_CONFIDENCE: 'thresholds.confidence.high',
```

```
  // Translation
  TRANSLATION_CONCURRENT_LIMIT: 'translation.concurrent_limit',
  TRANSLATION_PER_MINUTE_LIMIT: 'translation.per_minute_limit',

  // Workflow
  WORKFLOW_MIN_OCCURRENCES: 'workflow.proposal.min_occurrences',
  WORKFLOW_MIN_UNIQUE_USERS: 'workflow.proposal.min_unique_users',
} as const;

export type ConfigKey = typeof CONFIG_KEYS[keyof typeof CONFIG_KEYS];
```

42.5 CONFIGURATION SERVICE

File: packages/shared/src/config/ConfigurationService.ts

```
// ============================================================================
// RADIANT v4.8.0 - Configuration Service
// ============================================================================

import { Pool } from 'pg';
import Redis from 'ioredis';
import {
  SystemConfig,
  ResolvedConfig,
  ConfigValueType,
  ConfigUpdateRequest,
  TenantOverrideRequest,
  ConfigAuditEntry
} from './types';
import { CONFIG_KEYS } from './constants';

export class ConfigurationService {
  private pool: Pool;
  private redis: Redis | null;
  private localCache: Map<string, { value: ResolvedConfig; expiresAt: number }>;
  private defaultTtl: number = 300; // 5 minutes

  constructor(pool: Pool, redis?: Redis) {
    this.pool = pool;
    this.redis = redis || null;
    this.localCache = new Map();
  }

  /**
   * Get a configuration value with tenant override support
   */
  async get<T = any>(
```

```
  key: string,
  tenantId?: string,
  environment: string = 'prod'
): Promise<T> {
  const cacheKey = this.buildCacheKey(key, tenantId, environment);

  // Check local cache first
  const cached = this.localCache.get(cacheKey);
  if (cached && cached.expiresAt > Date.now()) {
    return cached.value.value as T;
  }

  // Check Redis cache
  if (this.redis) {
    const redisValue = await this.redis.get(cacheKey);
    if (redisValue) {
      const parsed = JSON.parse(redisValue) as ResolvedConfig;
      this.localCache.set(cacheKey, {
        value: parsed,
        expiresAt: Date.now() + this.defaultTtl * 1000
      });
      return parsed.value as T;
    }
  }

  // Fetch from database
  const result = await this.pool.query(
    `SELECT * FROM get_config($1, $2, $3)`,
    [key, tenantId, environment]
  );

  if (!result.rows[0]) {
    throw new Error(`Configuration not found: ${key}`);
  }

  const config = result.rows[0] as ResolvedConfig;
  const value = this.parseValue(config);

  // Update caches
  const resolvedConfig = { ...config, value };

  if (this.redis) {
    await this.redis.setex(cacheKey, this.defaultTtl, JSON.stringify(resolvedConfig));
  }

  this.localCache.set(cacheKey, {
    value: resolvedConfig,
    expiresAt: Date.now() + this.defaultTtl * 1000
```

```
  });

  return value as T;
}

/**
 * Get multiple configuration values
 */
async getMultiple(
  keys: string[],
  tenantId?: string,
  environment: string = 'prod'
): Promise<Record<string, any>> {
  const results: Record<string, any> = {};

  await Promise.all(
    keys.map(async (key) => {
      try {
        results[key] = await this.get(key, tenantId, environment);
      } catch {
        results[key] = undefined;
      }
    })
  );

  return results;
}

/**
 * Get all configurations in a category
 */
async getByCategory(
  categoryId: string,
  tenantId?: string,
  environment: string = 'prod'
): Promise<ResolvedConfig[]> {
  const result = await this.pool.query(
    `SELECT * FROM get_configs_by_category($1, $2, $3)`,
    [categoryId, tenantId, environment]
  );

  return result.rows;
}

/**
 * Update a global configuration value
 */
async update(
```

```
  key: string,
  request: ConfigUpdateRequest,
  updatedBy: string
): Promise<SystemConfig> {
  const config = await this.getConfigByKey(key);

  // Validate value
  this.validateValue(config, request.value);

  // Update based on type
  const updateColumn = this.getValueColumn(config.valueType);

  const result = await this.pool.query(`
    UPDATE system_configuration
    SET ${updateColumn} = $2,
        updated_at = NOW(),
        updated_by = $3
    WHERE key = $1
    RETURNING *
  `, [key, request.value, updatedBy]);

  // Invalidate caches
  await this.invalidateCache(key);

  return result.rows[0];
}

/**
 * Create a tenant-specific override
 */
async createTenantOverride(
  key: string,
  tenantId: string,
  request: TenantOverrideRequest,
  createdBy: string
): Promise<TenantConfigOverride> {
  const config = await this.getConfigByKey(key);

  // Validate value
  this.validateValue(config, request.value);

  const valueColumn = this.getValueColumn(config.valueType);

  const result = await this.pool.query(`
    INSERT INTO tenant_configuration_overrides (
      tenant_id, config_id, ${valueColumn}, valid_from, valid_until, reason, created_by
    ) VALUES ($1, $2, $3, COALESCE($4, NOW()), $5, $6, $7)
    ON CONFLICT (tenant_id, config_id) DO UPDATE SET
```

```
      ${valueColumn} = $3,
      valid_from = COALESCE($4, NOW()),
      valid_until = $5,
      reason = $6,
      updated_by = $7,
      updated_at = NOW()
    RETURNING *
  `, [tenantId, config.id, request.value, request.validFrom, request.validUntil, request.reas

  // Invalidate caches
  await this.invalidateCache(key, tenantId);

  return result.rows[0];
}

/**
 * Delete a tenant override (revert to global)
 */
async deleteTenantOverride(
  key: string,
  tenantId: string
): Promise<void> {
  const config = await this.getConfigByKey(key);

  await this.pool.query(`
    DELETE FROM tenant_configuration_overrides
    WHERE config_id = $1 AND tenant_id = $2
  `, [config.id, tenantId]);

  await this.invalidateCache(key, tenantId);
}

/**
 * Get audit log for a configuration
 */
async getAuditLog(
  key: string,
  options: { limit?: number; offset?: number; tenantId?: string } = {}
): Promise<ConfigAuditEntry[]> {
  const { limit = 50, offset = 0, tenantId } = options;

  let query = `
    SELECT * FROM configuration_audit_log
    WHERE config_key = $1
  `;
  const params: any[] = [key];

  if (tenantId) {
```

```
    query += ` AND (tenant_id = $${params.length + 1} OR tenant_id IS NULL)`;
    params.push(tenantId);
  }

  query += ` ORDER BY created_at DESC LIMIT $${params.length + 1} OFFSET $${params.length + 2}`;
  params.push(limit, offset);

  const result = await this.pool.query(query, params);
  return result.rows;
}

/**
 * Export all configurations
 */
async exportConfigs(tenantId?: string): Promise<ConfigExport> {
  const configs = await this.pool.query(`
    SELECT key,
           COALESCE(value_string, value_integer::text, value_decimal::text, value_boolean::text, value_
    FROM system_configuration
    WHERE is_deprecated = false
  `);

  let tenantOverrides: any[] = [];
  if (tenantId) {
    const overrides = await this.pool.query(`
      SELECT sc.key,
             COALESCE(tco.value_string, tco.value_integer::text, tco.value_decimal::text, tco.value_
             tco.reason
      FROM tenant_configuration_overrides tco
      JOIN system_configuration sc ON sc.id = tco.config_id
      WHERE tco.tenant_id = $1
    `, [tenantId]);

    tenantOverrides = overrides.rows.map(r => ({
      tenantId,
      key: r.key,
      value: r.value,
      reason: r.reason,
    }));
  }

  return {
    version: '4.8.0',
    exportedAt: new Date().toISOString(),
    exportedBy: 'system',
    configs: configs.rows,
    tenantOverrides: tenantOverrides.length > 0 ? tenantOverrides : undefined,
  };
```

```
}

// Private helpers

private buildCacheKey(key: string, tenantId?: string, environment?: string): string {
  return `config:${environment || 'prod'}:${tenantId || 'global'}:${key}`;
}

private async getConfigByKey(key: string): Promise<SystemConfig> {
  const result = await this.pool.query(
    `SELECT * FROM system_configuration WHERE key = $1`,
    [key]
  );

  if (!result.rows[0]) {
    throw new Error(`Configuration not found: ${key}`);
  }

  return result.rows[0];
}

private getValueColumn(type: ConfigValueType): string {
  switch (type) {
    case 'string':
    case 'enum':
      return 'value_string';
    case 'integer':
    case 'duration':
      return 'value_integer';
    case 'decimal':
    case 'percentage':
      return 'value_decimal';
    case 'boolean':
      return 'value_boolean';
    case 'json':
      return 'value_json';
    default:
      throw new Error(`Unknown value type: ${type}`);
  }
}

private parseValue(config: ResolvedConfig): any {
  const value = config.value;

  if (typeof value === 'string') {
    try {
      return JSON.parse(value);
    } catch {
```

```
      return value;
    }
  }

  return value;
}

private validateValue(config: SystemConfig, value: any): void {
  const numValue = typeof value === 'number' ? value : parseFloat(value);

  if (config.minValue !== undefined && numValue < config.minValue) {
    throw new Error(`Value must be at least ${config.minValue}`);
  }

  if (config.maxValue !== undefined && numValue > config.maxValue) {
    throw new Error(`Value must be at most ${config.maxValue}`);
  }

  if (config.enumValues && !config.enumValues.includes(String(value))) {
    throw new Error(`Value must be one of: ${config.enumValues.join(', ')}`);
  }

  if (config.regexPattern) {
    const regex = new RegExp(config.regexPattern);
    if (!regex.test(String(value))) {
      throw new Error(`Value does not match required pattern`);
    }
  }
}

private async invalidateCache(key: string, tenantId?: string): Promise<void> {
  // Clear local cache
  for (const cacheKey of this.localCache.keys()) {
    if (cacheKey.includes(key)) {
      this.localCache.delete(cacheKey);
    }
  }

  // Clear Redis cache
  if (this.redis) {
    const pattern = tenantId
      ? `config:*:${tenantId}:${key}`
      : `config:*:*:${key}`;

    const keys = await this.redis.keys(pattern);
    if (keys.length > 0) {
      await this.redis.del(...keys);
    }
```

```
    }

    // Add to invalidation queue for other instances
    await this.pool.query(`
      INSERT INTO configuration_cache_invalidation (config_key, tenant_id)
      VALUES ($1, $2)
    `, [key, tenantId]);
  }
}

// Singleton instance
let configService: ConfigurationService | null = null;

export function getConfigService(pool: Pool, redis?: Redis): ConfigurationService {
  if (!configService) {
    configService = new ConfigurationService(pool, redis);
  }
  return configService;
}

/**
 * Helper function to get a config value
 */
export async function getConfig<T = any>(
  key: string,
  tenantId?: string,
  environment?: string
): Promise<T> {
  if (!configService) {
    throw new Error('ConfigurationService not initialized');
  }
  return configService.get<T>(key, tenantId, environment);
}
```

42.6 ADMIN DASHBOARD - CONFIGURATION MANAGEMENT

File: admin-dashboard/app/configuration/page.tsx

```
// =============================================================================
// RADIANT v4.8.0 - Configuration Management Dashboard
// =============================================================================

'use client';

import React, { useState, useEffect } from 'react';
import {
  Box,
```

```
  Typography,
  Tabs,
  Tab,
  Card,
  CardContent,
  Table,
  TableBody,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  Paper,
  Chip,
  IconButton,
  Button,
  TextField,
  Dialog,
  DialogTitle,
  DialogContent,
  DialogActions,
  Switch,
  Slider,
  Select,
  MenuItem,
  Alert,
  Tooltip,
  InputAdornment,
  Collapse,
} from '@mui/material';
import {
  Edit,
  History,
  Refresh,
  Settings,
  Speed,
  Timer,
  AttachMoney,
  Token,
  Cached,
  TrendingUp,
  Discount,
  Lock,
  Translate,
  AccountTree,
  Notifications,
  ExpandMore,
  ExpandLess,
  Warning,
```

```
  Check,
} from '@mui/icons-material';
import { useTranslation } from '@/hooks/useTranslation';
import { api } from '@/lib/api';

interface ConfigCategory {
  id: string;
  name: string;
  description: string;
  icon: string;
}

interface SystemConfig {
  id: string;
  key: string;
  categoryId: string;
  valueType: string;
  value: any;
  displayName: string;
  description: string;
  unit: string;
  minValue: number;
  maxValue: number;
  enumValues: string[];
  isSensitive: boolean;
  requiresRestart: boolean;
  isOverride: boolean;
  updatedAt: string;
}

const CATEGORY_ICONS: Record<string, React.ReactNode> = {
  rate_limits: <Speed />,
  timeouts: <Timer />,
  pricing: <AttachMoney />,
  tokens: <Token />,
  retry: <Refresh />,
  cache: <Cached />,
  thresholds: <TrendingUp />,
  discounts: <Discount />,
  session: <Lock />,
  translation: <Translate />,
  workflow: <AccountTree />,
  notifications: <Notifications />,
};

export default function ConfigurationPage() {
  const { t } = useTranslation();
  const [categories, setCategories] = useState<ConfigCategory[]>([]);
```

```
const [activeCategory, setActiveCategory] = useState<string>('rate_limits');
const [configs, setConfigs] = useState<SystemConfig[]>([]);
const [editDialog, setEditDialog] = useState<SystemConfig | null>(null);
const [editValue, setEditValue] = useState<any>(null);
const [historyDialog, setHistoryDialog] = useState<string | null>(null);
const [auditLog, setAuditLog] = useState<any[]>([]);
const [loading, setLoading] = useState(true);
const [expandedKeys, setExpandedKeys] = useState<Set<string>>(new Set());

useEffect(() => {
  loadCategories();
}, []);

useEffect(() => {
  if (activeCategory) {
    loadConfigs(activeCategory);
  }
}, [activeCategory]);

async function loadCategories() {
  try {
    const res = await api.get('/configuration/categories');
    setCategories(res.data);
    if (res.data.length > 0) {
      setActiveCategory(res.data[0].id);
    }
  } catch (error) {
    console.error('Failed to load categories:', error);
  }
}

async function loadConfigs(categoryId: string) {
  setLoading(true);
  try {
    const res = await api.get(`/configuration/category/${categoryId}`);
    setConfigs(res.data);
  } catch (error) {
    console.error('Failed to load configs:', error);
  } finally {
    setLoading(false);
  }
}

async function handleSave() {
  if (!editDialog) return;
  try {
    await api.put(`/configuration/${editDialog.key}`, {
      value: editValue,
```

```
      reason: 'Updated via Admin Dashboard',
    });
    setEditDialog(null);
    loadConfigs(activeCategory);
  } catch (error) {
    console.error('Failed to save config:', error);
  }
}

async function loadAuditLog(key: string) {
  try {
    const res = await api.get(`/configuration/${key}/audit`);
    setAuditLog(res.data);
    setHistoryDialog(key);
  } catch (error) {
    console.error('Failed to load audit log:', error);
  }
}

function renderValueEditor(config: SystemConfig) {
  switch (config.valueType) {
    case 'boolean':
      return (
        <Switch
          checked={editValue === true || editValue === 'true'}
          onChange={(e) => setEditValue(e.target.checked)}
        />
      );

    case 'integer':
    case 'duration':
      return (
        <Box>
          <Slider
            value={Number(editValue)}
            onChange={(_, v) => setEditValue(v)}
            min={config.minValue || 0}
            max={config.maxValue || 100}
            valueLabelDisplay="auto"
          />
          <TextField
            type="number"
            value={editValue}
            onChange={(e) => setEditValue(Number(e.target.value))}
            InputProps={{
              endAdornment: config.unit && (
                <InputAdornment position="end">{config.unit}</InputAdornment>
              ),
```

```
        }}
        fullWidth
        sx={{ mt: 2 }}
      />
    </Box>
  );

case 'decimal':
case 'percentage':
  return (
    <Box>
      <Slider
        value={Number(editValue) * (config.valueType === 'percentage' ? 100 : 1)}
        onChange={(_, v) => setEditValue(
          (v as number) / (config.valueType === 'percentage' ? 100 : 1)
        )}
        min={(config.minValue || 0) * (config.valueType === 'percentage' ? 100 : 1)}
        max={(config.maxValue || 1) * (config.valueType === 'percentage' ? 100 : 1)}
        valueLabelDisplay="auto"
        valueLabelFormat={(v) => config.valueType === 'percentage' ? `${v}%` : v}
      />
      <TextField
        type="number"
        value={config.valueType === 'percentage' ? (editValue * 100).toFixed(0) : editVal
        onChange={(e) => {
          const val = Number(e.target.value);
          setEditValue(config.valueType === 'percentage' ? val / 100 : val);
        }}
        InputProps={{
          endAdornment: <InputAdornment position="end">%</InputAdornment>,
        }}
        fullWidth
        sx={{ mt: 2 }}
      />
    </Box>
  );

case 'enum':
  return (
    <Select
      value={editValue}
      onChange={(e) => setEditValue(e.target.value)}
      fullWidth
    >
      {config.enumValues?.map((val) => (
        <MenuItem key={val} value={val}>{val}</MenuItem>
      ))}
    </Select>
```

```
      );

    case 'json':
      return (
        <TextField
          multiline
          rows={6}
          value={typeof editValue === 'string' ? editValue : JSON.stringify(editValue, null,
          onChange={(e) => {
            try {
              setEditValue(JSON.parse(e.target.value));
            } catch {
              setEditValue(e.target.value);
            }
          }}
          fullWidth
          sx={{ fontFamily: 'monospace' }}
        />
      );

    default:
      return (
        <TextField
          value={editValue}
          onChange={(e) => setEditValue(e.target.value)}
          fullWidth
        />
      );
  }
}

function formatValue(config: SystemConfig): string {
  if (config.isSensitive) return '••••••••';

  switch (config.valueType) {
    case 'boolean':
      return config.value ? 'Enabled' : 'Disabled';
    case 'percentage':
      return `${(config.value * 100).toFixed(0)}%`;
    case 'duration':
      if (config.value >= 86400) return `${(config.value / 86400).toFixed(1)} days`;
      if (config.value >= 3600) return `${(config.value / 3600).toFixed(1)} hours`;
      if (config.value >= 60) return `${(config.value / 60).toFixed(0)} min`;
      return `${config.value} sec`;
    case 'json':
      return JSON.stringify(config.value);
    default:
      return `${config.value}${config.unit ? ` ${config.unit}` : ''}`;
```

```
  }
}

return (
  <Box sx={{ p: 3 }}>
    <Box sx={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center', mb: 3
      <Typography variant="h4">
        <Settings sx={{ mr: 1, verticalAlign: 'middle' }} />
        {t('admin.configuration.title')}
      </Typography>
      <Button
        variant="outlined"
        startIcon={<Refresh />}
        onClick={() => loadConfigs(activeCategory)}
      >
        {t('admin.buttons.refresh')}
      </Button>
    </Box>

    <Box sx={{ display: 'flex', gap: 3 }}>
      {/* Category Tabs (Vertical) */}
      <Paper sx={{ minWidth: 250 }}>
        <Tabs
          orientation="vertical"
          value={activeCategory}
          onChange={(_, v) => setActiveCategory(v)}
          sx={{ borderRight: 1, borderColor: 'divider' }}
        >
          {categories.map((cat) => (
            <Tab
              key={cat.id}
              value={cat.id}
              label={
                <Box sx={{ display: 'flex', alignItems: 'center', gap: 1 }}>
                  {CATEGORY_ICONS[cat.id]}
                  <span>{cat.name}</span>
                </Box>
              }
              sx={{ justifyContent: 'flex-start' }}
            />
          ))}
        </Tabs>
      </Paper>

      {/* Configuration Table */}
      <Box sx={{ flex: 1 }}>
        <TableContainer component={Paper}>
          <Table>
```

```
<TableHead>
  <TableRow>
    <TableCell>{t('admin.configuration.parameter')}</TableCell>
    <TableCell>{t('admin.configuration.value')}</TableCell>
    <TableCell>{t('admin.configuration.status')}</TableCell>
    <TableCell align="right">{t('admin.configuration.actions')}</TableCell>
  </TableRow>
</TableHead>
<TableBody>
  {configs.map((config) => (
    <React.Fragment key={config.key}>
      <TableRow hover>
        <TableCell>
          <Box>
            <Typography fontWeight="medium">
              {config.displayName}
              {config.requiresRestart && (
                <Tooltip title="Requires restart">
                  <Warning fontSize="small" color="warning" sx={{ ml: 1 }} />
                </Tooltip>
              )}
            </Typography>
            <Typography variant="caption" color="text.secondary">
              {config.key}
            </Typography>
          </Box>
        </TableCell>
        <TableCell>
          <Typography fontFamily="monospace">
            {formatValue(config)}
          </Typography>
        </TableCell>
        <TableCell>
          {config.isOverride ? (
            <Chip size="small" color="info" label="Override" />
          ) : (
            <Chip size="small" color="default" label="Default" />
          )}
        </TableCell>
        <TableCell align="right">
          <Tooltip title={t('admin.buttons.edit')}>
            <IconButton
              size="small"
              onClick={() => {
                setEditDialog(config);
                setEditValue(config.value);
              }}
            >
```

```
                    <Edit fontSize="small" />
                  </IconButton>
                </Tooltip>
                <Tooltip title={t('admin.configuration.history')}>
                  <IconButton
                    size="small"
                    onClick={() => loadAuditLog(config.key)}
                  >
                    <History fontSize="small" />
                  </IconButton>
                </Tooltip>
                <IconButton
                  size="small"
                  onClick={() => {
                    const newSet = new Set(expandedKeys);
                    if (newSet.has(config.key)) {
                      newSet.delete(config.key);
                    } else {
                      newSet.add(config.key);
                    }
                    setExpandedKeys(newSet);
                  }}
                >
                  {expandedKeys.has(config.key) ? <ExpandLess /> : <ExpandMore />}
                </IconButton>
              </TableCell>
            </TableRow>
            <TableRow>
              <TableCell colSpan={4} sx={{ py: 0 }}>
                <Collapse in={expandedKeys.has(config.key)}>
                  <Box sx={{ p: 2, bgcolor: 'grey.50' }}>
                    <Typography variant="body2" color="text.secondary">
                      {config.description}
                    </Typography>
                    {config.minValue !== null && (
                      <Typography variant="caption" display="block">
                        Min: {config.minValue} | Max: {config.maxValue}
                      </Typography>
                    )}
                  </Box>
                </Collapse>
              </TableCell>
            </TableRow>
          </React.Fragment>
        ))}
      </TableBody>
    </Table>
  </TableContainer>
```

```
      </Box>
    </Box>

    {/* Edit Dialog */}
    <Dialog open={!!editDialog} onClose={() => setEditDialog(null)} maxWidth="sm" fullWidth>
      <DialogTitle>
        {t('admin.configuration.edit_parameter')}
      </DialogTitle>
      <DialogContent>
        {editDialog && (
          <Box sx={{ pt: 2 }}>
            <Typography variant="subtitle2" gutterBottom>
              {editDialog.displayName}
            </Typography>
            <Typography variant="body2" color="text.secondary" gutterBottom>
              {editDialog.description}
            </Typography>
            <Box sx={{ mt: 3 }}>
              {renderValueEditor(editDialog)}
            </Box>
            {editDialog.requiresRestart && (
              <Alert severity="warning" sx={{ mt: 2 }}>
                {t('admin.configuration.requires_restart_warning')}
              </Alert>
            )}
          </Box>
        )}
      </DialogContent>
      <DialogActions>
        <Button onClick={() => setEditDialog(null)}>
          {t('admin.buttons.cancel')}
        </Button>
        <Button onClick={handleSave} variant="contained" color="primary">
          {t('admin.buttons.save')}
        </Button>
      </DialogActions>
    </Dialog>

    {/* History Dialog */}
    <Dialog open={!!historyDialog} onClose={() => setHistoryDialog(null)} maxWidth="md" fullW
      <DialogTitle>
        {t('admin.configuration.change_history')}: {historyDialog}
      </DialogTitle>
      <DialogContent>
        <TableContainer>
          <Table size="small">
            <TableHead>
              <TableRow>
```

```
                <TableCell>{t('admin.configuration.date')}</TableCell>
                <TableCell>{t('admin.configuration.action')}</TableCell>
                <TableCell>{t('admin.configuration.old_value')}</TableCell>
                <TableCell>{t('admin.configuration.new_value')}</TableCell>
                <TableCell>{t('admin.configuration.changed_by')}</TableCell>
              </TableRow>
            </TableHead>
            <TableBody>
              {auditLog.map((entry) => (
                <TableRow key={entry.id}>
                  <TableCell>
                    {new Date(entry.createdAt).toLocaleString()}
                  </TableCell>
                  <TableCell>
                    <Chip size="small" label={entry.action} />
                  </TableCell>
                  <TableCell>
                    <code>{JSON.stringify(entry.oldValue?.value)}</code>
                  </TableCell>
                  <TableCell>
                    <code>{JSON.stringify(entry.newValue?.value)}</code>
                  </TableCell>
                  <TableCell>{entry.changedByEmail || 'System'}</TableCell>
                </TableRow>
              ))}
            </TableBody>
          </Table>
        </TableContainer>
      </DialogContent>
      <DialogActions>
        <Button onClick={() => setHistoryDialog(null)}>
          {t('admin.buttons.close')}
        </Button>
      </DialogActions>
    </Dialog>
  </Box>
 );
}
```

42.7 API LAMBDA

File: `lambda/configuration/api.ts`

```
// ============================================================================
// RADIANT v4.8.0 - Configuration API Lambda
// ============================================================================
```

```typescript
import { APIGatewayProxyHandler, APIGatewayProxyResult } from 'aws-lambda';
import { Pool } from 'pg';
import { extractAuthContext, requireRoles } from '../shared/auth';
import { ConfigurationService } from '@radiant/shared/config';

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

const configService = new ConfigurationService(pool);

export const handler: APIGatewayProxyHandler = async (event): Promise<APIGatewayProxyResult> =>
  const { httpMethod, path, pathParameters, body } = event;

  try {
    const auth = extractAuthContext(event);
    requireRoles(auth, ['admin', 'super_admin']);

    // GET /configuration/categories
    if (httpMethod === 'GET' && path === '/configuration/categories') {
      return await getCategories();
    }

    // GET /configuration/category/:id
    if (httpMethod === 'GET' && path.match(/\/configuration\/category\/[a-z_]+$/)) {
      return await getByCategory(pathParameters?.id!, auth.tenantId);
    }

    // GET /configuration/:key
    if (httpMethod === 'GET' && path.match(/\/configuration\/[a-z_.]+$/)) {
      return await getConfig(pathParameters?.key!, auth.tenantId);
    }

    // PUT /configuration/:key
    if (httpMethod === 'PUT' && path.match(/\/configuration\/[a-z_.]+$/)) {
      requireRoles(auth, ['super_admin']);
      return await updateConfig(pathParameters?.key!, JSON.parse(body || '{}'), auth.userId);
    }

    // GET /configuration/:key/audit
    if (httpMethod === 'GET' && path.match(/\/configuration\/[a-z_.]+\/audit$/)) {
      return await getAuditLog(pathParameters?.key!, auth.tenantId);
    }

    // POST /configuration/:key/tenant-override
    if (httpMethod === 'POST' && path.match(/\/configuration\/[a-z_.]+\/tenant-override$/)) {
      requireRoles(auth, ['super_admin']);
```

```
      const { tenantId, ...rest } = JSON.parse(body || '{}');
      return await createTenantOverride(pathParameters?.key!, tenantId, rest, auth.userId);
    }

    // DELETE /configuration/:key/tenant-override/:tenantId
    if (httpMethod === 'DELETE' && path.match(/\/configuration\/[a-z_.]+\/tenant-override\/[a-
      requireRoles(auth, ['super_admin']);
      return await deleteTenantOverride(pathParameters?.key!, pathParameters?.tenantId!);
    }

    // POST /configuration/export
    if (httpMethod === 'POST' && path === '/configuration/export') {
      return await exportConfigs(auth.tenantId);
    }

    return { statusCode: 404, body: JSON.stringify({ error: 'Not found' }) };
  } catch (error) {
    console.error('Configuration API error:', error);
    return {
      statusCode: error.statusCode || 500,
      body: JSON.stringify({ error: error.message }),
    };
  }
};

async function getCategories(): Promise<APIGatewayProxyResult> {
  const result = await pool.query(`
    SELECT id, name, description, display_order, icon
    FROM configuration_categories
    ORDER BY display_order
  `);

  return {
    statusCode: 200,
    body: JSON.stringify(result.rows),
  };
}

async function getByCategory(categoryId: string, tenantId: string): Promise<APIGatewayProxyResu
  const configs = await configService.getByCategory(categoryId, tenantId);

  return {
    statusCode: 200,
    body: JSON.stringify(configs),
  };
}

async function getConfig(key: string, tenantId: string): Promise<APIGatewayProxyResult> {
```

```
  const value = await configService.get(key, tenantId);

  return {
    statusCode: 200,
    body: JSON.stringify({ key, value }),
  };
}

async function updateConfig(key: string, data: any, userId: string): Promise<APIGatewayProxyRes
  const config = await configService.update(key, data, userId);

  return {
    statusCode: 200,
    body: JSON.stringify(config),
  };
}

async function getAuditLog(key: string, tenantId: string): Promise<APIGatewayProxyResult> {
  const log = await configService.getAuditLog(key, { tenantId, limit: 100 });

  return {
    statusCode: 200,
    body: JSON.stringify(log),
  };
}

async function createTenantOverride(
  key: string,
  tenantId: string,
  data: any,
  userId: string
): Promise<APIGatewayProxyResult> {
  const override = await configService.createTenantOverride(key, tenantId, data, userId);

  return {
    statusCode: 201,
    body: JSON.stringify(override),
  };
}

async function deleteTenantOverride(key: string, tenantId: string): Promise<APIGatewayProxyResu
  await configService.deleteTenantOverride(key, tenantId);

  return {
    statusCode: 204,
    body: '',
  };
}
```

```
async function exportConfigs(tenantId: string): Promise<APIGatewayProxyResult> {
  const exportData = await configService.exportConfigs(tenantId);

  return {
    statusCode: 200,
    headers: {
      'Content-Type': 'application/json',
      'Content-Disposition': `attachment; filename="radiant-config-${new Date().toISOString().
    },
    body: JSON.stringify(exportData, null, 2),
  };
}
```

42.8 USAGE EXAMPLE - UPDATING CODE TO USE CONFIGURABLE VALUES

Before (Hardcoded):

```
// OLD CODE - Hardcoded values
const DEFAULT_MARGIN = 0.40;
const MAX_RETRY_ATTEMPTS = 3;
const SESSION_TIMEOUT = 1800;

async function calculateCost(providerCost: number): Promise<number> {
  return providerCost * (1 + DEFAULT_MARGIN);  // ← HARDCODED!
}

async function retryRequest(fn: () => Promise<any>): Promise<any> {
  for (let i = 0; i < MAX_RETRY_ATTEMPTS; i++) {  // ← HARDCODED!
    try {
      return await fn();
    } catch (error) {
      if (i === MAX_RETRY_ATTEMPTS - 1) throw error;
      await sleep(1000 * Math.pow(2, i));
    }
  }
}
```

After (Database-Driven):

```
// NEW CODE - Database-driven with ConfigurationService
import { getConfig, CONFIG_KEYS } from '@radiant/shared/config';

async function calculateCost(providerCost: number, tenantId: string): Promise<number> {
  const margin = await getConfig<number>(
    CONFIG_KEYS.EXTERNAL_PROVIDER_MARGIN,
```

```
    tenantId
  );
  return providerCost * (1 + margin);  //  CONFIGURABLE!
}

async function retryRequest(fn: () => Promise<any>, tenantId: string): Promise<any> {
  const [maxAttempts, initialDelay, backoffMultiplier] = await Promise.all([
    getConfig<number>(CONFIG_KEYS.MAX_RETRY_ATTEMPTS, tenantId),
    getConfig<number>(CONFIG_KEYS.INITIAL_RETRY_DELAY, tenantId),
    getConfig<number>(CONFIG_KEYS.BACKOFF_MULTIPLIER, tenantId),
  ]);

  for (let i = 0; i < maxAttempts; i++) {  //  CONFIGURABLE!
    try {
      return await fn();
    } catch (error) {
      if (i === maxAttempts - 1) throw error;
      await sleep(initialDelay * Math.pow(backoffMultiplier, i));
    }
  }
}
```

42.9 LOCALIZATION STRINGS FOR CONFIGURATION UI

Add to migration 041b_seed_localization.sql:

```
-- Configuration Management Strings
INSERT INTO localization_registry (key, default_text, category, context, source_app) VALUES
('admin.configuration.title', 'Configuration Management', 'features.admin', 'Page title', 'adm
('admin.configuration.parameter', 'Parameter', 'features.admin', 'Table header', 'admin'),
('admin.configuration.value', 'Value', 'features.admin', 'Table header', 'admin'),
('admin.configuration.status', 'Status', 'features.admin', 'Table header', 'admin'),
('admin.configuration.actions', 'Actions', 'features.admin', 'Table header', 'admin'),
('admin.configuration.history', 'History', 'features.admin', 'Button tooltip', 'admin'),
('admin.configuration.edit_parameter', 'Edit Parameter', 'features.admin', 'Dialog title', 'adm
('admin.configuration.change_history', 'Change History', 'features.admin', 'Dialog title', 'adm
('admin.configuration.date', 'Date', 'features.admin', 'Table header', 'admin'),
('admin.configuration.action', 'Action', 'features.admin', 'Table header', 'admin'),
('admin.configuration.old_value', 'Old Value', 'features.admin', 'Table header', 'admin'),
('admin.configuration.new_value', 'New Value', 'features.admin', 'Table header', 'admin'),
('admin.configuration.changed_by', 'Changed By', 'features.admin', 'Table header', 'admin'),
('admin.configuration.requires_restart_warning', 'This change requires a service restart to tal
```

IMPLEMENTATION VERIFICATION CHECKLIST

Complete v4.8.0 Verification

Section 42: Dynamic Configuration Management System

- ☐ Migration 042_configuration_management.sql applied successfully
- ☐ Migration 042b_seed_configuration.sql applied with all parameters
- ☐ configuration_categories table created with 12 categories
- ☐ system_configuration table created with type validation
- ☐ tenant_configuration_overrides table created with RLS
- ☐ configuration_audit_log table created
- ☐ configuration_cache_invalidation table created
- ☐ get_config() function working with tenant override support
- ☐ get_configs_by_category() function returning configs
- ☐ Audit log triggers capturing all changes
- ☐ Cache invalidation triggers working
- ☐ ConfigurationService TypeScript class implemented
- ☐ getConfig() helper function working with caching
- ☐ Redis caching layer integrated
- ☐ Configuration API Lambda deployed
- ☐ Admin /configuration page accessible
- ☐ Category tabs displaying all 12 categories
- ☐ Edit dialog with type-appropriate editors
- ☐ History dialog showing audit log
- ☐ Tenant override creation working
- ☐ Configuration export/import working
- ☐ All hardcoded values replaced with getConfig() calls
- ☐ Localization strings added for configuration UI

Section 41: Complete Internationalization System (from v4.7.0)

- ☐ All previous v4.7.0 checklist items verified

Section 40: Application-Level Data Isolation (from v4.6.0)

- ☐ All previous v4.6.0 checklist items verified

Integration Verification

- ☐ All Lambda functions using ConfigurationService
- ☐ No hardcoded rate limits remaining
- ☐ No hardcoded timeouts remaining
- ☐ No hardcoded pricing margins remaining
- ☐ No hardcoded token limits remaining
- ☐ No hardcoded retry configurations remaining

☐ Configuration changes propagate without deployment
☐ Per-tenant overrides working correctly
☐ Audit trail complete for all changes
