Contents

SECTION 37: FEEDBACK LEARNING SYSTEM & NEURAL ENGINE LOOP (v4.3.0)

> **Dependencies:** Sections 0-36, especially 11 (Brain) and 13 (Neural Engine) **Creates:** Complete feedback loop from user signals → Neural Engine → Brain decisions

---

37.1 Feedback System Overview

The Feedback Learning System creates a closed-loop where user feedback continuously improves AI routing decisions. The system captures both explicit feedback (thumbs up/down) and implicit signals (regenerate, copy, abandon), ties each to a complete execution manifest, and feeds everything into the Neural Engine which advises the Brain.

Architecture Flow

```
                          FEEDBACK LEARNING LOOP


  User Request
```

```
                       RADIANT BRAIN
• Consults Neural Engine for recommendations
• Considers user/tenant/global learning
• Applies confidence thresholds
• Selects orchestration → services → models



                   EXECUTION MANIFEST
Records: output_id, models[], versions[], orchestration_id,
         services[], thermal_states{}, provider_health{},
         brain_reasoning, latency_ms, cost, timestamp



                       AI RESPONSE
Delivered to user with output_id for feedback reference



 EXPLICIT             IMPLICIT                     VOICE
   + cat          regenerate                    any lang
+ text               copy/share                   transcribe
              abandon                        translate
                      switch model



                     NEURAL ENGINE
• Aggregates feedback by model/orchestration/service
• Updates model scores (individual → tenant → global)
• Applies confidence thresholds and decay
• Generates routing recommendations
• Tracks A/B experiment outcomes



                  BRAIN INTELLIGENCE
Real-time: Neural recommendations inform next Brain decision
```

```
Batch: Nightly aggregation updates routing weights
```

Learning Scope Hierarchy

```
                    TIERED LEARNING SCOPE

Priority 1: USER SCOPE (Most Personalized)

  • Alice's feedback improves Alice's experience
  • Fast adaptation to individual preferences
  • Requires minimum ~20 feedback samples for confidence


Priority 2: TENANT SCOPE (Organization-Wide)

  • Company X's employees collectively train Company X's Brain
  • Isolated from Company Y (privacy boundary)
  • Captures organizational preferences (e.g., "we prefer Claude")


Priority 3: GLOBAL SCOPE (Platform-Wide, Anonymized)

  • Aggregate patterns: "Claude wins 73% for legal tasks"
  • Cold start defaults for new users/tenants
  • Configurable opt-in/opt-out per tenant
```

37.2 Feedback Database Schema

```sql
-- migrations/037_feedback_learning_system.sql

-- ============================================================================
-- EXECUTION MANIFESTS: Full provenance for every AI output
-- ============================================================================

CREATE TABLE execution_manifests (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    output_id VARCHAR(100) UNIQUE NOT NULL,  -- Reference ID for feedback
    -- Context
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    conversation_id UUID REFERENCES thinktank_conversations(id),
    message_id UUID REFERENCES thinktank_messages(id),

    -- What was requested
    request_type VARCHAR(50) NOT NULL,  -- 'chat', 'completion', 'orchestration', 'service'
    task_type VARCHAR(50),  -- 'code', 'creative', 'analysis', 'medical', etc.
    domain_mode VARCHAR(50),  -- Think Tank domain mode if applicable
    input_prompt_hash VARCHAR(64),  -- SHA-256 of input for deduplication
    input_tokens INTEGER,
    input_language VARCHAR(10),  -- Detected input language (ISO 639-1)

    -- What was used (THE MANIFEST)
    models_used TEXT[] NOT NULL,
    model_versions JSONB DEFAULT '{}',  -- {"claude-sonnet-4": "20241022", ...}
    orchestration_id UUID REFERENCES workflow_definitions(id),
    orchestration_name VARCHAR(100),
    services_used TEXT[],  -- ['perception', 'medical', ...]
    thermal_states_at_execution JSONB DEFAULT '{}',  -- {"whisper-large-v3": "HOT", ...}
    provider_health_at_execution JSONB DEFAULT '{}',  -- {"anthropic": {"latency_ms": 150, "er

    -- Brain's decision
    brain_reasoning TEXT,  -- Why Brain chose this path
    brain_confidence DECIMAL(3, 2),  -- 0.00-1.00
    was_user_override BOOLEAN DEFAULT false,  -- User manually selected model

    -- Outcome metrics
    output_tokens INTEGER,
    total_latency_ms INTEGER,
    time_to_first_token_ms INTEGER,
    total_cost DECIMAL(10, 6),
    was_streamed BOOLEAN DEFAULT false,

    -- For multi-step orchestrations
    step_count INTEGER DEFAULT 1,
    step_details JSONB DEFAULT '[]',  -- [{model, latency, tokens, cost}, ...]

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT valid_confidence CHECK (brain_confidence >= 0 AND brain_confidence <= 1)
);

CREATE INDEX idx_exec_manifest_output ON execution_manifests(output_id);
CREATE INDEX idx_exec_manifest_tenant_user ON execution_manifests(tenant_id, user_id);
CREATE INDEX idx_exec_manifest_conversation ON execution_manifests(conversation_id);
CREATE INDEX idx_exec_manifest_models ON execution_manifests USING GIN(models_used);
CREATE INDEX idx_exec_manifest_task ON execution_manifests(task_type);
CREATE INDEX idx_exec_manifest_created ON execution_manifests(created_at DESC);

-- ============================================================================
-- EXPLICIT FEEDBACK: Thumbs up/down with optional categories
-- ============================================================================

CREATE TYPE feedback_rating AS ENUM ('positive', 'negative', 'neutral');
CREATE TYPE feedback_category AS ENUM (
    'accuracy',        -- Factually correct
    'relevance',       -- Answered the question
    'tone',            -- Appropriate style
    'format',          -- Good structure/formatting
    'speed',           -- Fast enough
    'safety',          -- Appropriate content
    'creativity',      -- Novel/interesting
    'completeness',    -- Fully addressed request
    'other'
);

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

    -- Link to execution
    output_id VARCHAR(100) NOT NULL REFERENCES execution_manifests(output_id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- The feedback
    rating feedback_rating NOT NULL,
    categories feedback_category[] DEFAULT '{}',  -- What specifically was good/bad
    comment_text TEXT,  -- Optional text feedback
    comment_language VARCHAR(10),  -- Detected language of comment

    -- Metadata
    feedback_source VARCHAR(50) DEFAULT 'thinktank',  -- 'thinktank', 'api', 'admin'
    user_agent TEXT,
    client_timestamp TIMESTAMPTZ,  -- When user clicked

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    -- One feedback per output per user
    UNIQUE(output_id, user_id)
);

CREATE INDEX idx_feedback_explicit_output ON feedback_explicit(output_id);
CREATE INDEX idx_feedback_explicit_tenant ON feedback_explicit(tenant_id);
CREATE INDEX idx_feedback_explicit_rating ON feedback_explicit(rating);
CREATE INDEX idx_feedback_explicit_created ON feedback_explicit(created_at DESC);

-- ============================================================================
-- IMPLICIT FEEDBACK: Behavioral signals (higher volume, weighted less)
-- ============================================================================

CREATE TYPE implicit_signal_type AS ENUM (
    'regenerate',            -- User clicked regenerate (negative)
    'copy_response',         -- User copied response (positive)
    'share_response',        -- User shared response (very positive)
    'export_response',       -- User exported (positive)
    'continue_conversation',-- User sent follow-up (neutral-positive)
    'abandon_conversation', -- User left without follow-up (weak negative)
    'abandon_mid_response', -- User stopped streaming (negative)
    'manual_model_switch',  -- User changed model (negative for current)
    'edit_and_resend',       -- User edited their prompt (neutral)
    'response_time_short',  -- Quick reply = engaged (positive)
    'response_time_long',   -- Slow reply = confused/distracted (neutral)
    'scroll_to_end',         -- Read full response (positive)
    'scroll_bounce',         -- Scrolled away quickly (negative)
    'favorite_added',        -- Added to favorites (very positive)
    'report_content'         -- Reported for review (very negative)
);

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

    -- Link to execution
    output_id VARCHAR(100) NOT NULL REFERENCES execution_manifests(output_id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- The signal
    signal_type implicit_signal_type NOT NULL,
    signal_value JSONB DEFAULT '{}',  -- Additional context (e.g., time_to_next_message_ms)

    -- Computed sentiment score (-1.0 to +1.0)
    sentiment_score DECIMAL(3, 2) NOT NULL,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_feedback_implicit_output ON feedback_implicit(output_id);
CREATE INDEX idx_feedback_implicit_tenant ON feedback_implicit(tenant_id);
CREATE INDEX idx_feedback_implicit_signal ON feedback_implicit(signal_type);
CREATE INDEX idx_feedback_implicit_created ON feedback_implicit(created_at DESC);
```

```sql
-- ============================================================================
-- VOICE FEEDBACK: Multi-language voice input with transcription
-- ============================================================================

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

    -- Link to execution
    output_id VARCHAR(100) NOT NULL REFERENCES execution_manifests(output_id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- Audio storage
    audio_s3_key VARCHAR(500) NOT NULL,
    audio_duration_seconds DECIMAL(8, 2),
    audio_format VARCHAR(20),  -- 'webm', 'mp3', 'wav', etc.

    -- Transcription
    transcription_text TEXT,
    original_language VARCHAR(10),  -- Detected language (ISO 639-1)
    translated_text TEXT,  -- English translation if not English
    transcription_confidence DECIMAL(3, 2),
    transcription_model VARCHAR(50),  -- 'whisper-large-v3', etc.

    -- Sentiment analysis of voice feedback
    sentiment_score DECIMAL(3, 2),  -- -1.0 to +1.0
    inferred_rating feedback_rating,
    inferred_categories feedback_category[],

    -- Processing status
    processing_status VARCHAR(20) DEFAULT 'pending',  -- 'pending', 'processing', 'completed',
    processing_error TEXT,
    processed_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_feedback_voice_output ON feedback_voice(output_id);
CREATE INDEX idx_feedback_voice_status ON feedback_voice(processing_status);

-- ============================================================================
-- NEURAL MODEL SCORES: Learned effectiveness per model/task/scope
-- ============================================================================

CREATE TYPE learning_scope AS ENUM ('user', 'tenant', 'global');

CREATE TABLE neural_model_scores (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    -- Scope (null = global)
    scope learning_scope NOT NULL,
    tenant_id UUID REFERENCES tenants(id),  -- null for global
    user_id UUID REFERENCES users(id),      -- null for tenant/global

    -- What we're scoring
    model_id VARCHAR(100) NOT NULL,
    task_type VARCHAR(50),  -- null = overall score for model
    domain_mode VARCHAR(50),  -- null = all domains

    -- Aggregated scores (0.0 to 1.0)
    effectiveness_score DECIMAL(4, 3) NOT NULL DEFAULT 0.500,
    accuracy_score DECIMAL(4, 3),
    relevance_score DECIMAL(4, 3),
    speed_score DECIMAL(4, 3),

    -- Statistics
    positive_count INTEGER DEFAULT 0,
    negative_count INTEGER DEFAULT 0,
    neutral_count INTEGER DEFAULT 0,
    implicit_positive_count INTEGER DEFAULT 0,
    implicit_negative_count INTEGER DEFAULT 0,
    total_feedback_count INTEGER DEFAULT 0,

    -- Confidence in this score (based on sample size)
    confidence DECIMAL(3, 2) DEFAULT 0.00,

    -- Model version tracking
    last_model_version VARCHAR(50),
    score_decay_applied_at TIMESTAMPTZ,  -- When we last decayed old scores

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    -- Unique per scope/model/task combination
    UNIQUE NULLS NOT DISTINCT (scope, tenant_id, user_id, model_id, task_type, domain_mode)
);

CREATE INDEX idx_neural_scores_model ON neural_model_scores(model_id);
CREATE INDEX idx_neural_scores_scope ON neural_model_scores(scope);
CREATE INDEX idx_neural_scores_tenant ON neural_model_scores(tenant_id);
CREATE INDEX idx_neural_scores_effectiveness ON neural_model_scores(effectiveness_score DESC);

-- ============================================================================
-- NEURAL ROUTING RECOMMENDATIONS: Brain reads these for decisions
-- ============================================================================
CREATE TABLE neural_routing_recommendations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    -- Scope
    scope learning_scope NOT NULL,
    tenant_id UUID REFERENCES tenants(id),
    user_id UUID REFERENCES users(id),

    -- Recommendation context
    task_type VARCHAR(50) NOT NULL,
    domain_mode VARCHAR(50),
    input_characteristics JSONB DEFAULT '{}',  -- {requires_vision, requires_audio, token_esti

    -- The recommendation
    recommended_model VARCHAR(100) NOT NULL,
    recommended_orchestration_id UUID REFERENCES workflow_definitions(id),
    recommended_services TEXT[],

    -- Alternatives (ordered by score)
    alternative_models TEXT[],

    -- Confidence
    recommendation_confidence DECIMAL(3, 2) NOT NULL,
    sample_size INTEGER,  -- How much data this is based on

    -- Reasoning (for debugging/transparency)
    reasoning TEXT,

    -- Validity
    valid_from TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    valid_until TIMESTAMPTZ,  -- null = no expiry
    is_active BOOLEAN DEFAULT true,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_neural_rec_active ON neural_routing_recommendations(is_active, scope);
CREATE INDEX idx_neural_rec_task ON neural_routing_recommendations(task_type, domain_mode);
CREATE INDEX idx_neural_rec_tenant ON neural_routing_recommendations(tenant_id);

-- ============================================================================
-- USER TRUST SCORES: Anti-gaming feedback weighting
-- ============================================================================

CREATE TABLE user_trust_scores (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- Trust factors
    trust_score DECIMAL(3, 2) NOT NULL DEFAULT 0.50,  -- 0.00-1.00
    account_age_days INTEGER,
    total_feedback_count INTEGER DEFAULT 0,
    feedback_diversity_score DECIMAL(3, 2),  -- How varied their feedback is
    feedback_alignment_score DECIMAL(3, 2),  -- How aligned with population

    -- Flags
    is_outlier BOOLEAN DEFAULT false,
    outlier_reason TEXT,
    is_rate_limited BOOLEAN DEFAULT false,
    rate_limit_until TIMESTAMPTZ,

    -- Last update
    last_feedback_at TIMESTAMPTZ,
    last_trust_calculation_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(tenant_id, user_id)
);

CREATE INDEX idx_trust_scores_tenant ON user_trust_scores(tenant_id);
CREATE INDEX idx_trust_scores_outlier ON user_trust_scores(is_outlier);

-- ============================================================================
-- A/B TESTING: Measure if routing changes improve outcomes
-- ============================================================================

CREATE TYPE experiment_status AS ENUM ('draft', 'running', 'paused', 'completed', 'cancelled')

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

    -- Experiment definition
    name VARCHAR(200) NOT NULL,
    description TEXT,
    hypothesis TEXT,

    -- Scope
    tenant_id UUID REFERENCES tenants(id),  -- null = all tenants

    -- What we're testing
    experiment_type VARCHAR(50) NOT NULL,  -- 'model_routing', 'orchestration', 'service'
    control_config JSONB NOT NULL,  -- The current/default behavior
    treatment_config JSONB NOT NULL,  -- The new behavior being tested

    -- Traffic allocation
    traffic_percentage DECIMAL(5, 2) DEFAULT 10.00,  -- % of users in treatment

    -- Status
    status experiment_status NOT NULL DEFAULT 'draft',
    started_at TIMESTAMPTZ,
    ended_at TIMESTAMPTZ,

    -- Results
    control_sample_size INTEGER DEFAULT 0,
    treatment_sample_size INTEGER DEFAULT 0,
    control_positive_rate DECIMAL(5, 4),
    treatment_positive_rate DECIMAL(5, 4),
    statistical_significance DECIMAL(5, 4),  -- p-value
    effect_size DECIMAL(5, 4),  -- Cohen's d
    winner VARCHAR(20),  -- 'control', 'treatment', 'inconclusive'

    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_ab_experiments_status ON ab_experiments(status);
CREATE INDEX idx_ab_experiments_tenant ON ab_experiments(tenant_id);

CREATE TABLE ab_experiment_assignments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    experiment_id UUID NOT NULL REFERENCES ab_experiments(id) ON DELETE CASCADE,
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- Assignment
    variant VARCHAR(20) NOT NULL,  -- 'control' or 'treatment'
    assigned_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(experiment_id, user_id)
);

CREATE INDEX idx_ab_assignments_experiment ON ab_experiment_assignments(experiment_id);
CREATE INDEX idx_ab_assignments_user ON ab_experiment_assignments(user_id);

-- ============================================================================
-- ROW LEVEL SECURITY
-- ============================================================================

ALTER TABLE execution_manifests ENABLE ROW LEVEL SECURITY;
ALTER TABLE feedback_explicit ENABLE ROW LEVEL SECURITY;
ALTER TABLE feedback_implicit ENABLE ROW LEVEL SECURITY;
ALTER TABLE feedback_voice ENABLE ROW LEVEL SECURITY;
ALTER TABLE neural_model_scores ENABLE ROW LEVEL SECURITY;
ALTER TABLE neural_routing_recommendations ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_trust_scores ENABLE ROW LEVEL SECURITY;
ALTER TABLE ab_experiments ENABLE ROW LEVEL SECURITY;
ALTER TABLE ab_experiment_assignments ENABLE ROW LEVEL SECURITY;

CREATE POLICY exec_manifest_isolation ON execution_manifests
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY feedback_explicit_isolation ON feedback_explicit
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY feedback_implicit_isolation ON feedback_implicit
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY feedback_voice_isolation ON feedback_voice
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY neural_scores_isolation ON neural_model_scores
    USING (tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY neural_rec_isolation ON neural_routing_recommendations
    USING (tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY trust_scores_isolation ON user_trust_scores
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY ab_experiments_isolation ON ab_experiments
    USING (tenant_id IS NULL OR tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY ab_assignments_isolation ON ab_experiment_assignments
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

-- ===========================================================================
-- HELPER FUNCTIONS
-- ===========================================================================

-- Get aggregated feedback score for an output
CREATE OR REPLACE FUNCTION get_feedback_score(p_output_id VARCHAR)
RETURNS JSONB AS $$
DECLARE
    explicit_score DECIMAL;
    implicit_score DECIMAL;
    combined_score DECIMAL;
    result JSONB;
BEGIN
    -- Get explicit feedback
    SELECT
        CASE rating
            WHEN 'positive' THEN 1.0
            WHEN 'negative' THEN -1.0
            ELSE 0.0
        END INTO explicit_score
    FROM feedback_explicit
    WHERE output_id = p_output_id
    LIMIT 1;

    -- Get average implicit feedback
    SELECT AVG(sentiment_score) INTO implicit_score
    FROM feedback_implicit
    WHERE output_id = p_output_id;

    -- Combine (explicit weighted 3x)
    combined_score := COALESCE(
        (COALESCE(explicit_score, 0) * 3 + COALESCE(implicit_score, 0)) /
        CASE
            WHEN explicit_score IS NOT NULL AND implicit_score IS NOT NULL THEN 4
            WHEN explicit_score IS NOT NULL THEN 3
            WHEN implicit_score IS NOT NULL THEN 1
            ELSE 1
        END,
        0
    );

    result := jsonb_build_object(
        'explicit_score', explicit_score,
        'implicit_score', implicit_score,
        'combined_score', combined_score,
        'has_explicit', explicit_score IS NOT NULL,
        'has_implicit', implicit_score IS NOT NULL
    );

    RETURN result;
END;
$$ LANGUAGE plpgsql;

-- Get best model recommendation for context
CREATE OR REPLACE FUNCTION get_neural_recommendation(
    p_tenant_id UUID,
    p_user_id UUID,
    p_task_type VARCHAR,
    p_domain_mode VARCHAR DEFAULT NULL
)
RETURNS TABLE (
    model_id VARCHAR,
    confidence DECIMAL,
    scope learning_scope,
    reasoning TEXT
) AS $$
BEGIN
    -- Try user-specific first, then tenant, then global
    RETURN QUERY
    SELECT
        r.recommended_model,
        r.recommendation_confidence,
        r.scope,
        r.reasoning
    FROM neural_routing_recommendations r
    WHERE r.is_active = true
    AND (r.valid_until IS NULL OR r.valid_until > NOW())
    AND r.task_type = p_task_type
    AND (r.domain_mode IS NULL OR r.domain_mode = p_domain_mode)
    AND (
        (r.scope = 'user' AND r.tenant_id = p_tenant_id AND r.user_id = p_user_id) OR
        (r.scope = 'tenant' AND r.tenant_id = p_tenant_id AND r.user_id IS NULL) OR
        (r.scope = 'global' AND r.tenant_id IS NULL AND r.user_id IS NULL)
    )
    ORDER BY
        CASE r.scope
            WHEN 'user' THEN 1
            WHEN 'tenant' THEN 2
            WHEN 'global' THEN 3
        END,
        r.recommendation_confidence DESC
    LIMIT 1;
END;
$$ LANGUAGE plpgsql;

-- Calculate implicit signal sentiment
CREATE OR REPLACE FUNCTION get_implicit_sentiment(p_signal_type implicit_signal_type)
RETURNS DECIMAL AS $$
BEGIN
    RETURN CASE p_signal_type
        WHEN 'copy_response' THEN 0.7
        WHEN 'share_response' THEN 0.9
        WHEN 'export_response' THEN 0.6
        WHEN 'favorite_added' THEN 0.9
        WHEN 'continue_conversation' THEN 0.3
        WHEN 'scroll_to_end' THEN 0.2
        WHEN 'response_time_short' THEN 0.2
        WHEN 'regenerate' THEN -0.8
        WHEN 'manual_model_switch' THEN -0.6
        WHEN 'abandon_conversation' THEN -0.3
        WHEN 'abandon_mid_response' THEN -0.7
        WHEN 'scroll_bounce' THEN -0.4
        WHEN 'report_content' THEN -1.0
        WHEN 'edit_and_resend' THEN 0.0
        WHEN 'response_time_long' THEN 0.0
        ELSE 0.0
    END;
END;
$$ LANGUAGE plpgsql IMMUTABLE;
```

37.3 Shared Types for Feedback System

```
// packages/shared/src/types/feedback.types.ts

/**
 * RADIANT v4.3.0 - Feedback System Types
 */

// ============================================================================
// EXECUTION MANIFEST
// ============================================================================

export interface ExecutionManifest {
    id: string;
    outputId: string;  // Unique reference for feedback

    // Context
    tenantId: string;
    userId: string;
    conversationId?: string;
    messageId?: string;

    // Request
    requestType: 'chat' | 'completion' | 'orchestration' | 'service';
    taskType?: TaskType;
    domainMode?: DomainMode;
    inputPromptHash?: string;
    inputTokens?: number;
    inputLanguage?: string;

    // THE MANIFEST - What was used
    modelsUsed: string[];
    modelVersions: Record<string, string>;
    orchestrationId?: string;
    orchestrationName?: string;
    servicesUsed: string[];
    thermalStatesAtExecution: Record<string, ThermalState>;
    providerHealthAtExecution: Record<string, ProviderHealthSnapshot>;

    // Brain decision
    brainReasoning?: string;
    brainConfidence: number;
    wasUserOverride: boolean;

    // Outcome
    outputTokens?: number;
    totalLatencyMs: number;
    timeToFirstTokenMs?: number;
    totalCost: number;
    wasStreamed: boolean;

    // Multi-step
    stepCount: number;
    stepDetails: ExecutionStep[];

    createdAt: Date;
}

export interface ExecutionStep {
    stepIndex: number;
    modelId: string;
    latencyMs: number;
    inputTokens: number;
    outputTokens: number;
    cost: number;
}

export interface ProviderHealthSnapshot {
    latencyMs: number;
    errorRate: number;
    status: 'healthy' | 'degraded' | 'unhealthy';
}

export type TaskType =
    | 'chat'
    | 'code'
    | 'analysis'
    | 'creative'
    | 'vision'
    | 'audio'
    | 'medical'
    | 'legal'
    | 'research'
    | 'translation';

export type DomainMode =
    | 'general'
    | 'medical'
    | 'legal'
    | 'code'
    | 'creative'
    | 'research'
    | 'business';

// ============================================================================
// EXPLICIT FEEDBACK
// ============================================================================

export type FeedbackRating = 'positive' | 'negative' | 'neutral';

export type FeedbackCategory =
    | 'accuracy'
    | 'relevance'
    | 'tone'
    | 'format'
    | 'speed'
    | 'safety'
    | 'creativity'
    | 'completeness'
    | 'other';

export interface ExplicitFeedback {
    id: string;
    outputId: string;
    tenantId: string;
    userId: string;

    rating: FeedbackRating;
    categories: FeedbackCategory[];
    commentText?: string;
    commentLanguage?: string;

    feedbackSource: 'thinktank' | 'api' | 'admin';
    createdAt: Date;
}

export interface FeedbackSubmission {
    outputId: string;
    rating: FeedbackRating;
    categories?: FeedbackCategory[];
    commentText?: string;
}

// ============================================================================
// IMPLICIT FEEDBACK
// ============================================================================

export type ImplicitSignalType =
    | 'regenerate'
    | 'copy_response'
    | 'share_response'
    | 'export_response'
    | 'continue_conversation'
    | 'abandon_conversation'
    | 'abandon_mid_response'
    | 'manual_model_switch'
    | 'edit_and_resend'
    | 'response_time_short'
    | 'response_time_long'
    | 'scroll_to_end'
    | 'scroll_bounce'
    | 'favorite_added'
    | 'report_content';

export interface ImplicitFeedback {
    id: string;
    outputId: string;
    tenantId: string;
    userId: string;

    signalType: ImplicitSignalType;
    signalValue: Record<string, unknown>;
    sentimentScore: number;  // -1.0 to +1.0

    createdAt: Date;
}

export interface ImplicitSignalSubmission {
    outputId: string;
    signalType: ImplicitSignalType;
    signalValue?: Record<string, unknown>;
}

// ============================================================================
// VOICE FEEDBACK
// ============================================================================

export interface VoiceFeedback {
    id: string;
    outputId: string;
    tenantId: string;
    userId: string;

    audioS3Key: string;
    audioDurationSeconds: number;
    audioFormat: string;
    transcriptionText?: string;
    originalLanguage?: string;
    translatedText?: string;
    transcriptionConfidence?: number;
    transcriptionModel?: string;

    sentimentScore?: number;
    inferredRating?: FeedbackRating;
    inferredCategories?: FeedbackCategory[];

    processingStatus: 'pending' | 'processing' | 'completed' | 'failed';
    processedAt?: Date;

    createdAt: Date;
}

// ============================================================================
// NEURAL LEARNING
// ============================================================================

export type LearningScope = 'user' | 'tenant' | 'global';

export interface NeuralModelScore {
    id: string;
    scope: LearningScope;
    tenantId?: string;
    userId?: string;

    modelId: string;
    taskType?: TaskType;
    domainMode?: DomainMode;

    effectivenessScore: number;  // 0.0-1.0
    accuracyScore?: number;
    relevanceScore?: number;
    speedScore?: number;

    positiveCount: number;
    negativeCount: number;
    neutralCount: number;
    implicitPositiveCount: number;
    implicitNegativeCount: number;
    totalFeedbackCount: number;

    confidence: number;  // 0.0-1.0

    updatedAt: Date;
```

```
}

export interface NeuralRecommendation {
    id: string;
    scope: LearningScope;
    tenantId?: string;
    userId?: string;

    taskType: TaskType;
    domainMode?: DomainMode;
    inputCharacteristics: Record<string, unknown>;

    recommendedModel: string;
    recommendedOrchestrationId?: string;
    recommendedServices: string[];
    alternativeModels: string[];

    recommendationConfidence: number;
    sampleSize: number;
    reasoning: string;

    isActive: boolean;
    validFrom: Date;
    validUntil?: Date;
}

// ============================================================================
// TRUST & ANTI-GAMING
// ============================================================================

export interface UserTrustScore {
    id: string;
    tenantId: string;
    userId: string;

    trustScore: number;  // 0.0-1.0
    accountAgeDays: number;
    totalFeedbackCount: number;
    feedbackDiversityScore: number;
    feedbackAlignmentScore: number;

    isOutlier: boolean;
    outlierReason?: string;
    isRateLimited: boolean;
    rateLimitUntil?: Date;

    updatedAt: Date;
}
```

```
// ============================================================================
// A/B TESTING
// ============================================================================

export type ExperimentStatus = 'draft' | 'running' | 'paused' | 'completed' | 'cancelled';

export interface ABExperiment {
    id: string;
    name: string;
    description?: string;
    hypothesis?: string;

    tenantId?: string;
    experimentType: 'model_routing' | 'orchestration' | 'service';
    controlConfig: Record<string, unknown>;
    treatmentConfig: Record<string, unknown>;

    trafficPercentage: number;
    status: ExperimentStatus;

    controlSampleSize: number;
    treatmentSampleSize: number;
    controlPositiveRate?: number;
    treatmentPositiveRate?: number;
    statisticalSignificance?: number;
    effectSize?: number;
    winner?: 'control' | 'treatment' | 'inconclusive';

    startedAt?: Date;
    endedAt?: Date;
}

// Import existing types
import { ThermalState } from './ai.types';
```

Update the shared types index:

```
// packages/shared/src/types/index.ts

// Add to existing exports
export * from './feedback.types';
```

37.4 API Endpoints Summary

Feedback Endpoints

Method	Endpoint	Description
POST	`/api/v2/feedback/explicit`	Submit thumbs up/down with optional categories
POST	`/api/v2/feedback/implicit`	Record implicit signal (regenerate, copy, etc.)
POST	`/api/v2/feedback/voice`	Upload voice feedback (multipart)
GET	`/api/v2/feedback/stats/{output_id}`	Get aggregated feedback for output

Neural Engine Endpoints

Method	Endpoint	Description
GET	`/api/v2/neural/recommendation`	Get Neural Engine recommendation
GET	`/api/v2/neural/scores`	Get model scores for context
POST	`/api/v2/neural/learn`	Trigger learning for output (internal)

Voice Processing Endpoints

Method	Endpoint	Description
POST	`/api/v2/voice/transcribe`	Transcribe audio to text (any language)
POST	`/api/v2/voice/translate`	Translate text to English

Service Layer (Client Apps)

Method	Endpoint	Description
GET	`/api/v2/service/feedback/config`	Get feedback widget configuration
POST	`/api/v2/service/feedback/implicit/batch`	Batch submit implicit signals
GET	`/api/v2/service/feedback/conversation/{id}/summary`	Get conversation feedback summary

Admin Endpoints

Method	Endpoint	Description
GET	`/api/v2/admin/feedback/stats`	Feedback analytics dashboard
GET	`/api/v2/admin/experiments`	List A/B experiments
POST	`/api/v2/admin/experiments`	Create A/B experiment
PUT	`/api/v2/admin/experiments/{id}`	Update experiment status
GET	`/api/v2/admin/trust-scores`	View user trust scores

37.5 Implicit Signal Sentiment Mapping

Signal Type	Sentiment Score	Interpretation
`favorite_added`	+0.9	Very positive - user values this response
`share_response`	+0.9	Very positive - worth sharing
`copy_response`	+0.7	Positive - useful enough to copy
`export_response`	+0.6	Positive - keeping for later
`continue_conversation`	+0.3	Neutral-positive - engaged
`scroll_to_end`	+0.2	Weak positive - read full response
`response_time_short`	+0.2	Weak positive - quick engagement
`edit_and_resend`	0.0	Neutral - refining question
`response_time_long`	0.0	Neutral - thinking/distracted
`abandon_conversation`	-0.3	Weak negative - may be done or frustrated
`scroll_bounce`	-0.4	Negative - didn't read response
`manual_model_switch`	-0.6	Negative - current model wasn't working
`abandon_mid_response`	-0.7	Negative - stopped streaming early
`regenerate`	-0.8	Negative - response wasn't good
`report_content`	-1.0	Very negative - safety/quality issue

37.6 Learning Weighting Formula

Combined feedback score for model scoring:

```
weighted_score = (
    (explicit_score × 3.0 × trust_weight) +
    (implicit_score × 1.0) +
    (voice_score × 2.0 × trust_weight)
) / total_weight
```

Where: - `explicit_score`: -1.0 to +1.0 from thumbs rating - `implicit_score`: Average of implicit signal sentiments - `voice_score`: Sentiment analysis of transcribed voice feedback - `trust_weight`: 0.1 to 1.0 based on user trust score

Model effectiveness update:

```
new_score = (current_score × current_count + weighted_score) / (current_count + 1)
confidence = min(total_count / min_sample_size, 1.0)
```

Summary

RADIANT v4.3.0 (PROMPT-17) adds **Feedback Learning System & Neural Engine Loop**:

Section 37: Feedback Learning System (v4.3.0)

1. **Database Schema** (037_feedback_learning_system.sql)
   - `execution_manifests` - Full provenance for every AI output
   - `feedback_explicit` - Thumbs up/down with categories
   - `feedback_implicit` - Behavioral signals (regenerate, copy, abandon, etc.)
   - `feedback_voice` - Multi-language voice feedback with transcription
   - `neural_model_scores` - Learned effectiveness per model/task/scope
   - `neural_routing_recommendations` - Brain advice from Neural Engine
   - `user_trust_scores` - Anti-gaming trust levels
   - `ab_experiments` - A/B testing experiments
2. **Learning Scopes**
   - User-level: Personal preferences, fastest adaptation
   - Tenant-level: Organization-wide patterns, privacy isolated
   - Global-level: Platform defaults, cold start handling
3. **Signal Types**
   - Explicit: Thumbs up/down + 8 feedback categories + text comments
   - Implicit: 15 behavioral signals (regenerate, copy, share, abandon, etc.)
   - Voice: Multi-language voice feedback with Whisper transcription
4. **Neural Engine Integration**
   - Brain consults Neural Engine before every routing decision
   - Neural scores weighted at 35% in model selection
   - Real-time + nightly batch learning
5. **Anti-Gaming Protection**
   - User trust scores (0.0-1.0)
   - Rate limiting (50 feedback/hour)
   - Outlier detection (deviation from population)
   - Account age weighting
6. **A/B Testing Framework**
   - Create experiments comparing routing strategies
   - Automatic user assignment to control/treatment
   - Statistical significance tracking
