AI-Based Automated STEM Evaluation System - Database Schema Design

1. Database Overview

1.1 Database Information

- **Database**: PostgreSQL 15+
- **Character Set**: UTF8
- **Collation**: en_US.UTF-8
- **Timezone**: UTC
- **Naming Convention**: snake_case
- **Primary Keys**: UUID (using gen_random_uuid())

1.2 Design Principles

- **Normalization**: 3NF (Third Normal Form)
- **Soft Deletes**: Use `deleted_at` column instead of hard deletes
- **Audit Trails**: Track `created_at`, `updated_at`, `created_by`, `updated_by`
- **JSONB Usage**: For flexible schema elements (metadata, settings)
- **Partitioning**: For large tables (submissions, logs)
- **Indexes**: Strategic indexing for performance

1.3 Schema Diagram Overview

```
organizations
      users
          sessions
          password_reset_tokens
          oauth_connections
          parent_student_links
      classes
          class_students
          class_coaches
          teams
              team_members
      assessments
          assessment_assignments
          assessment_sessions
              assessment_responses
      submissions
          submission_files
          evaluations
      forum_posts
          forum_comments
          forum_votes
      notifications
```

2. Core Tables

2.1 organizations

Organizations represent schools, districts, or robotics programs.

```
CREATE TABLE organizations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    slug VARCHAR(100) UNIQUE NOT NULL,
    type VARCHAR(50) NOT NULL, -- 'school', 'district', 'club'
    address TEXT,
    city VARCHAR(100),
    state VARCHAR(50),
    country VARCHAR(50) DEFAULT 'USA',
    postal_code VARCHAR(20),
    phone VARCHAR(20),
    email VARCHAR(255),
    website VARCHAR(255),
    settings JSONB DEFAULT '{}',
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_organizations_slug ON organizations(slug);
CREATE INDEX idx_organizations_type ON organizations(type);
CREATE INDEX idx_organizations_deleted_at ON organizations(deleted_at);
```

Indexes Rationale: - `slug`: Lookup by organization slug in URLs - `type`: Filter organizations by type - `deleted_at`: Support soft deletes

---

2.2 users

Central user table for all user types (students, coaches, parents, admins).

```
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID NOT NULL REFERENCES organizations(id),
    email VARCHAR(255) UNIQUE NOT NULL,
    email_verified BOOLEAN DEFAULT false,
    email_verified_at TIMESTAMP WITH TIME ZONE,
    password_hash VARCHAR(255), -- nullable for OAuth-only users
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    display_name VARCHAR(200),
```

```
    role VARCHAR(20) NOT NULL, -- 'student', 'coach', 'parent', 'admin'
    avatar_url TEXT,
    bio TEXT,
    phone VARCHAR(20),
    date_of_birth DATE,
    grade_level VARCHAR(20), -- for students

    -- Preferences
    notification_preferences JSONB DEFAULT '{
        "email_enabled": true,
        "submission_feedback": true,
        "assessment_assigned": true,
        "forum_replies": true,
        "forum_mentions": true,
        "weekly_digest": false,
        "digest_day": "monday"
    }',

    -- Privacy
    profile_visibility VARCHAR(20) DEFAULT 'class', -- 'public', 'class', 'private'

    -- Account status
    is_active BOOLEAN DEFAULT true,
    last_login_at TIMESTAMP WITH TIME ZONE,
    login_count INTEGER DEFAULT 0,

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE,

    CONSTRAINT valid_role CHECK (role IN ('student', 'coach', 'parent', 'admin')),
    CONSTRAINT valid_profile_visibility CHECK (profile_visibility IN ('public', 'class', 'pr
);

CREATE INDEX idx_users_organization_id ON users(organization_id);
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_role ON users(role);
CREATE INDEX idx_users_deleted_at ON users(deleted_at);
CREATE INDEX idx_users_organization_role ON users(organization_id, role);
```

Indexes Rationale: - `organization_id`: Filter users by organization - `email`: Login and user lookup - `role`: Filter users by role - `organization_id, role`: Common query pattern (e.g., all students in org)

2.3 sessions

Active user sessions for authentication.

```
CREATE TABLE sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    refresh_token_hash VARCHAR(255) NOT NULL UNIQUE,
    access_token_jti VARCHAR(255), -- JWT ID for access token
    ip_address INET,
    user_agent TEXT,
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    last_activity_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_sessions_user_id ON sessions(user_id);
CREATE INDEX idx_sessions_expires_at ON sessions(expires_at);
CREATE INDEX idx_sessions_refresh_token_hash ON sessions(refresh_token_hash);
```

2.4 password_reset_tokens

Temporary tokens for password reset flow.

```
CREATE TABLE password_reset_tokens (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    token_hash VARCHAR(255) NOT NULL UNIQUE,
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL,
    used_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_password_reset_tokens_user_id ON password_reset_tokens(user_id);
CREATE INDEX idx_password_reset_tokens_token_hash ON password_reset_tokens(token_hash);
CREATE INDEX idx_password_reset_tokens_expires_at ON password_reset_tokens(expires_at);
```

2.5 oauth_connections

OAuth provider connections (Google, etc.).

```
CREATE TABLE oauth_connections (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    provider VARCHAR(50) NOT NULL, -- 'google', 'microsoft'
```

```
    provider_user_id VARCHAR(255) NOT NULL,
    provider_email VARCHAR(255),
    access_token TEXT,
    refresh_token TEXT,
    expires_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(provider, provider_user_id)
);

CREATE INDEX idx_oauth_connections_user_id ON oauth_connections(user_id);
CREATE INDEX idx_oauth_connections_provider ON oauth_connections(provider, provider_user_id)
```

3. Class and Team Management

3.1 classes

Classes managed by coaches.

```
CREATE TABLE classes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID NOT NULL REFERENCES organizations(id),
    name VARCHAR(255) NOT NULL,
    description TEXT,
    grade_level VARCHAR(50),
    school_year VARCHAR(20), -- '2025-2026'
    invite_code VARCHAR(10) UNIQUE NOT NULL,
    settings JSONB DEFAULT '{}',
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE,
    created_by UUID REFERENCES users(id)
);

CREATE INDEX idx_classes_organization_id ON classes(organization_id);
CREATE INDEX idx_classes_invite_code ON classes(invite_code);
CREATE INDEX idx_classes_deleted_at ON classes(deleted_at);
```

3.2 class_students

Many-to-many relationship between classes and students.

```
CREATE TABLE class_students (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    class_id UUID NOT NULL REFERENCES classes(id) ON DELETE CASCADE,
    student_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    joined_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) DEFAULT 'active', -- 'active', 'inactive', 'withdrawn'

    UNIQUE(class_id, student_id),
    CONSTRAINT student_role CHECK (
        (SELECT role FROM users WHERE id = student_id) = 'student'
    )
);

CREATE INDEX idx_class_students_class_id ON class_students(class_id);
CREATE INDEX idx_class_students_student_id ON class_students(student_id);
CREATE INDEX idx_class_students_status ON class_students(status);
```

3.3 class_coaches

Many-to-many relationship between classes and coaches.

```
CREATE TABLE class_coaches (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    class_id UUID NOT NULL REFERENCES classes(id) ON DELETE CASCADE,
    coach_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    role VARCHAR(20) DEFAULT 'instructor', -- 'instructor', 'assistant'
    assigned_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(class_id, coach_id),
    CONSTRAINT coach_role CHECK (
        (SELECT role FROM users WHERE id = coach_id) = 'coach'
    )
);

CREATE INDEX idx_class_coaches_class_id ON class_coaches(class_id);
CREATE INDEX idx_class_coaches_coach_id ON class_coaches(coach_id);
```

3.4 teams

VEX robotics teams.

```
CREATE TABLE teams (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID NOT NULL REFERENCES organizations(id),
    class_id UUID REFERENCES classes(id),
    name VARCHAR(255) NOT NULL,
```

```
    team_number VARCHAR(50), -- Official VEX team number like '12345A'
    competition_type VARCHAR(20) NOT NULL, -- 'VEX_IQ', 'VEX_V5'
    description TEXT,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE,
    created_by UUID REFERENCES users(id),

    CONSTRAINT valid_competition_type CHECK (competition_type IN ('VEX_IQ', 'VEX_V5'))
);

CREATE INDEX idx_teams_organization_id ON teams(organization_id);
CREATE INDEX idx_teams_class_id ON teams(class_id);
CREATE INDEX idx_teams_team_number ON teams(team_number);
CREATE INDEX idx_teams_deleted_at ON teams(deleted_at);
```

3.5 team_members

Students in teams with roles.

```
CREATE TABLE team_members (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    team_id UUID NOT NULL REFERENCES teams(id) ON DELETE CASCADE,
    student_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    role VARCHAR(50), -- 'driver', 'programmer', 'builder', 'designer', 'notebook'
    is_captain BOOLEAN DEFAULT false,
    joined_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    left_at TIMESTAMP WITH TIME ZONE,

    UNIQUE(team_id, student_id),
    CONSTRAINT student_role CHECK (
        (SELECT role FROM users WHERE id = student_id) = 'student'
    )
);

CREATE INDEX idx_team_members_team_id ON team_members(team_id);
CREATE INDEX idx_team_members_student_id ON team_members(student_id);
```

3.6 parent_student_links

Parent-student account relationships.

```sql
CREATE TABLE parent_student_links (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    parent_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    student_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    relationship VARCHAR(50), -- 'parent', 'guardian', 'mentor'
    status VARCHAR(20) DEFAULT 'pending', -- 'pending', 'approved', 'rejected'
    requested_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    approved_at TIMESTAMP WITH TIME ZONE,
    approved_by UUID REFERENCES users(id),

    UNIQUE(parent_id, student_id),
    CONSTRAINT parent_role CHECK (
        (SELECT role FROM users WHERE id = parent_id) = 'parent'
    ),
    CONSTRAINT student_role CHECK (
        (SELECT role FROM users WHERE id = student_id) = 'student'
    ),
    CONSTRAINT no_self_link CHECK (parent_id != student_id)
);

CREATE INDEX idx_parent_student_links_parent_id ON parent_student_links(parent_id);
CREATE INDEX idx_parent_student_links_student_id ON parent_student_links(student_id);
CREATE INDEX idx_parent_student_links_status ON parent_student_links(status);
```

4. Assessment System

4.1 questions

Question bank for assessments.

```sql
CREATE TABLE questions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID REFERENCES organizations(id),
    created_by UUID NOT NULL REFERENCES users(id),
    question_type VARCHAR(50) NOT NULL, -- 'multiple_choice', 'short_answer', 'code', 'essay
    difficulty_level INTEGER DEFAULT 2, -- 1 (easy) to 5 (hard)
    category VARCHAR(100), -- 'programming', 'engineering', 'problem_solving'
    tags TEXT[],

    -- Question content
    question_text TEXT NOT NULL,
    question_html TEXT,
    question_metadata JSONB DEFAULT '{}',

    -- For multiple choice
    options JSONB, -- [{"id": "a", "text": "...", "is_correct": true}, ...]
```

```
    -- For short answer / code
    expected_answer TEXT,
    evaluation_rubric TEXT,

    -- Points
    points INTEGER DEFAULT 1,

    -- Usage stats
    times_used INTEGER DEFAULT 0,
    avg_correctness DECIMAL(5,2),

    -- Versioning
    version INTEGER DEFAULT 1,
    parent_question_id UUID REFERENCES questions(id),

    -- Status
    is_active BOOLEAN DEFAULT true,
    is_public BOOLEAN DEFAULT false,

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE,

    CONSTRAINT valid_question_type CHECK (question_type IN ('multiple_choice', 'short_answer
    CONSTRAINT valid_difficulty CHECK (difficulty_level BETWEEN 1 AND 5)
);

CREATE INDEX idx_questions_organization_id ON questions(organization_id);
CREATE INDEX idx_questions_created_by ON questions(created_by);
CREATE INDEX idx_questions_type ON questions(question_type);
CREATE INDEX idx_questions_difficulty ON questions(difficulty_level);
CREATE INDEX idx_questions_category ON questions(category);
CREATE INDEX idx_questions_tags ON questions USING GIN(tags);
CREATE INDEX idx_questions_deleted_at ON questions(deleted_at);
```

4.2 question_pools

Reusable question collections.

```
CREATE TABLE question_pools (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID REFERENCES organizations(id),
    created_by UUID NOT NULL REFERENCES users(id),
    name VARCHAR(255) NOT NULL,
```

```
    description TEXT,
    tags TEXT[],
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_question_pools_organization_id ON question_pools(organization_id);
CREATE INDEX idx_question_pools_created_by ON question_pools(created_by);
```

4.3 question_pool_items

Questions in a pool.

```
CREATE TABLE question_pool_items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES question_pools(id) ON DELETE CASCADE,
    question_id UUID NOT NULL REFERENCES questions(id) ON DELETE CASCADE,
    sequence_order INTEGER,
    weight DECIMAL(5,2) DEFAULT 1.0,
    added_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(pool_id, question_id)
);

CREATE INDEX idx_question_pool_items_pool_id ON question_pool_items(pool_id);
CREATE INDEX idx_question_pool_items_question_id ON question_pool_items(question_id);
```

4.4 assessments

Assessment definitions created by coaches.

```
CREATE TABLE assessments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID NOT NULL REFERENCES organizations(id),
    created_by UUID NOT NULL REFERENCES users(id),
    title VARCHAR(255) NOT NULL,
    description TEXT,
    question_pool_id UUID REFERENCES question_pools(id),

    -- Configuration
    question_count INTEGER NOT NULL,
    difficulty_adaptation BOOLEAN DEFAULT false,
```

```
    time_limit_minutes INTEGER,
    passing_score INTEGER,
    max_attempts INTEGER DEFAULT 1,
    show_correct_answers BOOLEAN DEFAULT false,
    show_results_immediately BOOLEAN DEFAULT true,
    randomize_questions BOOLEAN DEFAULT true,

    -- Metadata
    tags TEXT[],

    -- Status
    is_published BOOLEAN DEFAULT false,

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_assessments_organization_id ON assessments(organization_id);
CREATE INDEX idx_assessments_created_by ON assessments(created_by);
CREATE INDEX idx_assessments_pool_id ON assessments(question_pool_id);
CREATE INDEX idx_assessments_deleted_at ON assessments(deleted_at);
```

4.5 assessment_assignments

Assignments of assessments to students/classes.

```
CREATE TABLE assessment_assignments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    assessment_id UUID NOT NULL REFERENCES assessments(id) ON DELETE CASCADE,
    target_type VARCHAR(20) NOT NULL, -- 'student', 'class', 'team'
    target_id UUID NOT NULL, -- ID of student, class, or team
    assigned_by UUID NOT NULL REFERENCES users(id),

    -- Scheduling
    available_from TIMESTAMP WITH TIME ZONE,
    due_date TIMESTAMP WITH TIME ZONE,

    -- Status
    status VARCHAR(20) DEFAULT 'assigned', -- 'assigned', 'in_progress', 'completed', 'overd

    -- Audit
    assigned_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
```

```
    CONSTRAINT valid_target_type CHECK (target_type IN ('student', 'class', 'team'))
);

CREATE INDEX idx_assessment_assignments_assessment_id ON assessment_assignments(assessment_i
CREATE INDEX idx_assessment_assignments_target ON assessment_assignments(target_type, target
CREATE INDEX idx_assessment_assignments_status ON assessment_assignments(status);
CREATE INDEX idx_assessment_assignments_due_date ON assessment_assignments(due_date);
```

4.6 assessment_sessions

Individual student assessment attempts.

```
CREATE TABLE assessment_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    assessment_id UUID NOT NULL REFERENCES assessments(id),
    student_id UUID NOT NULL REFERENCES users(id),
    assignment_id UUID REFERENCES assessment_assignments(id),

    -- Session state
    status VARCHAR(20) DEFAULT 'in_progress', -- 'in_progress', 'completed', 'abandoned', '
    current_question_index INTEGER DEFAULT 0,

    -- Selected questions (for this session)
    question_ids UUID[] NOT NULL,

    -- Timing
    started_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP WITH TIME ZONE,
    expires_at TIMESTAMP WITH TIME ZONE,
    time_spent_seconds INTEGER DEFAULT 0,

    -- Results
    total_questions INTEGER NOT NULL,
    answered_questions INTEGER DEFAULT 0,
    correct_answers INTEGER DEFAULT 0,
    score INTEGER,
    passed BOOLEAN,

    -- Metadata
    ip_address INET,
    user_agent TEXT,

    CONSTRAINT valid_status CHECK (status IN ('in_progress', 'completed', 'abandoned', 'expi
    CONSTRAINT student_role CHECK (
        (SELECT role FROM users WHERE id = student_id) = 'student'
```

```
    )
);

CREATE INDEX idx_assessment_sessions_assessment_id ON assessment_sessions(assessment_id);
CREATE INDEX idx_assessment_sessions_student_id ON assessment_sessions(student_id);
CREATE INDEX idx_assessment_sessions_status ON assessment_sessions(status);
CREATE INDEX idx_assessment_sessions_completed_at ON assessment_sessions(completed_at);
```

4.7 assessment_responses

Individual question responses in a session.

```
CREATE TABLE assessment_responses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES assessment_sessions(id) ON DELETE CASCADE,
    question_id UUID NOT NULL REFERENCES questions(id),
    sequence_number INTEGER NOT NULL,

    -- Response
    answer_text TEXT,
    answer_option_id VARCHAR(10), -- For multiple choice
    answer_code TEXT, -- For coding questions
    answer_metadata JSONB DEFAULT '{}',

    -- Evaluation
    is_correct BOOLEAN,
    points_earned INTEGER DEFAULT 0,
    points_possible INTEGER NOT NULL,

    -- AI evaluation (for subjective questions)
    ai_evaluation JSONB,
    ai_feedback TEXT,
    manual_override BOOLEAN DEFAULT false,
    evaluated_by UUID REFERENCES users(id),

    -- Timing
    answered_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    evaluated_at TIMESTAMP WITH TIME ZONE,
    time_spent_seconds INTEGER,

    UNIQUE(session_id, question_id)
);

CREATE INDEX idx_assessment_responses_session_id ON assessment_responses(session_id);
CREATE INDEX idx_assessment_responses_question_id ON assessment_responses(question_id);
```

5. Submission and Evaluation System

5.1 submissions

Student work submissions.

```
CREATE TABLE submissions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    student_id UUID NOT NULL REFERENCES users(id),
    class_id UUID REFERENCES classes(id),
    team_id UUID REFERENCES teams(id),

    -- Submission details
    title VARCHAR(255) NOT NULL,
    description TEXT,
    submission_type VARCHAR(50) NOT NULL, -- 'robot_image', 'code', 'notebook', 'mixed'
    competition_type VARCHAR(20), -- 'VEX_IQ', 'VEX_V5'

    -- Status
    status VARCHAR(20) DEFAULT 'uploaded', -- 'uploaded', 'processing', 'evaluated', 'failed

    -- Results
    overall_score INTEGER,

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE,

    CONSTRAINT valid_submission_type CHECK (submission_type IN ('robot_image', 'code', 'note
    CONSTRAINT valid_status CHECK (status IN ('uploaded', 'processing', 'evaluated', 'failed
    CONSTRAINT valid_competition_type CHECK (competition_type IN ('VEX_IQ', 'VEX_V5', NULL))
);

CREATE INDEX idx_submissions_student_id ON submissions(student_id);
CREATE INDEX idx_submissions_class_id ON submissions(class_id);
CREATE INDEX idx_submissions_team_id ON submissions(team_id);
CREATE INDEX idx_submissions_status ON submissions(status);
CREATE INDEX idx_submissions_created_at ON submissions(created_at DESC);
CREATE INDEX idx_submissions_deleted_at ON submissions(deleted_at);
```

5.2 submission_files

Files attached to submissions.

```sql
CREATE TABLE submission_files (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    submission_id UUID NOT NULL REFERENCES submissions(id) ON DELETE CASCADE,

    -- File details
    filename VARCHAR(255) NOT NULL,
    original_filename VARCHAR(255) NOT NULL,
    file_type VARCHAR(50) NOT NULL, -- 'image', 'code', 'document'
    mime_type VARCHAR(100) NOT NULL,
    file_size_bytes BIGINT NOT NULL,

    -- Storage
    storage_path TEXT NOT NULL,
    storage_bucket VARCHAR(100) NOT NULL,
    public_url TEXT,
    thumbnail_url TEXT,

    -- Processing
    is_processed BOOLEAN DEFAULT false,
    processing_error TEXT,

    -- Metadata
    file_metadata JSONB DEFAULT '{}',

    -- Audit
    uploaded_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_submission_files_submission_id ON submission_files(submission_id);
CREATE INDEX idx_submission_files_file_type ON submission_files(file_type);
```

5.3 evaluations

AI-generated evaluations for submissions.

```sql
CREATE TABLE evaluations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    submission_id UUID NOT NULL REFERENCES submissions(id) ON DELETE CASCADE,

    -- Overall evaluation
    overall_score INTEGER CHECK (overall_score BETWEEN 0 AND 100),
    feedback_summary TEXT,

    -- Detailed scores
```

```
    scores JSONB DEFAULT '{}', -- {"design": 85, "programming": 90, "documentation": 80}

    -- AI-generated content
    strengths TEXT[],
    improvements TEXT[],
    detailed_feedback TEXT,

    -- Per-file evaluations
    file_evaluations JSONB DEFAULT '[]',

    -- Code-specific
    code_analysis JSONB,
    optimization_suggestions JSONB,

    -- Image-specific
    image_analysis JSONB,
    components_identified TEXT[],

    -- Notebook-specific
    notebook_analysis JSONB,
    rubric_scores JSONB,

    -- AI metadata
    ai_model VARCHAR(100),
    ai_model_version VARCHAR(50),
    prompt_version VARCHAR(50),
    ai_cost DECIMAL(10,4),
    processing_time_ms INTEGER,
    confidence_score DECIMAL(5,2),

    -- Status
    status VARCHAR(20) DEFAULT 'pending', -- 'pending', 'completed', 'failed'
    error_message TEXT,

    -- Manual review
    reviewed_by UUID REFERENCES users(id),
    review_notes TEXT,
    reviewed_at TIMESTAMP WITH TIME ZONE,

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP WITH TIME ZONE,

    CONSTRAINT valid_status CHECK (status IN ('pending', 'completed', 'failed'))
);
```

```
CREATE INDEX idx_evaluations_submission_id ON evaluations(submission_id);
CREATE INDEX idx_evaluations_status ON evaluations(status);
CREATE INDEX idx_evaluations_created_at ON evaluations(created_at);
```

6. Forum System

6.1 forum_posts

Discussion board posts.

```
CREATE TABLE forum_posts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    author_id UUID NOT NULL REFERENCES users(id),
    organization_id UUID REFERENCES organizations(id),
    class_id UUID REFERENCES classes(id), -- Optional: restrict to class

    -- Content
    title VARCHAR(500) NOT NULL,
    content TEXT NOT NULL,
    content_html TEXT,

    -- Post type
    is_question BOOLEAN DEFAULT false,
    is_answered BOOLEAN DEFAULT false,
    accepted_answer_id UUID, -- References forum_comments(id)

    -- Categorization
    tags TEXT[],
    category VARCHAR(100),

    -- Engagement
    vote_count INTEGER DEFAULT 0,
    comment_count INTEGER DEFAULT 0,
    view_count INTEGER DEFAULT 0,

    -- Moderation
    is_pinned BOOLEAN DEFAULT false,
    is_locked BOOLEAN DEFAULT false,
    is_hidden BOOLEAN DEFAULT false,
    moderation_notes TEXT,
    moderated_by UUID REFERENCES users(id),
    moderated_at TIMESTAMP WITH TIME ZONE,

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE,
```

```
    edited_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_forum_posts_author_id ON forum_posts(author_id);
CREATE INDEX idx_forum_posts_organization_id ON forum_posts(organization_id);
CREATE INDEX idx_forum_posts_class_id ON forum_posts(class_id);
CREATE INDEX idx_forum_posts_tags ON forum_posts USING GIN(tags);
CREATE INDEX idx_forum_posts_is_question ON forum_posts(is_question);
CREATE INDEX idx_forum_posts_is_answered ON forum_posts(is_answered) WHERE is_question = tru
CREATE INDEX idx_forum_posts_created_at ON forum_posts(created_at DESC);
CREATE INDEX idx_forum_posts_vote_count ON forum_posts(vote_count DESC);
CREATE INDEX idx_forum_posts_deleted_at ON forum_posts(deleted_at);
```

6.2 forum_comments

Comments and replies on posts.

```
CREATE TABLE forum_comments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    post_id UUID NOT NULL REFERENCES forum_posts(id) ON DELETE CASCADE,
    author_id UUID NOT NULL REFERENCES users(id),
    parent_comment_id UUID REFERENCES forum_comments(id) ON DELETE CASCADE,

    -- Content
    content TEXT NOT NULL,
    content_html TEXT,

    -- Answer status
    is_accepted_answer BOOLEAN DEFAULT false,

    -- Engagement
    vote_count INTEGER DEFAULT 0,
    reply_count INTEGER DEFAULT 0,

    -- Threading
    depth_level INTEGER DEFAULT 0,
    thread_path TEXT, -- Materialized path like '1.2.3'

    -- Moderation
    is_hidden BOOLEAN DEFAULT false,
    moderation_notes TEXT,
    moderated_by UUID REFERENCES users(id),
    moderated_at TIMESTAMP WITH TIME ZONE,

    -- Audit
```

```
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE,
    edited_at TIMESTAMP WITH TIME ZONE,

    CONSTRAINT max_depth CHECK (depth_level <= 5)
);

CREATE INDEX idx_forum_comments_post_id ON forum_comments(post_id);
CREATE INDEX idx_forum_comments_author_id ON forum_comments(author_id);
CREATE INDEX idx_forum_comments_parent_id ON forum_comments(parent_comment_id);
CREATE INDEX idx_forum_comments_created_at ON forum_comments(created_at);
CREATE INDEX idx_forum_comments_is_accepted ON forum_comments(is_accepted_answer);
CREATE INDEX idx_forum_comments_deleted_at ON forum_comments(deleted_at);
```

6.3 forum_votes

User votes on posts and comments.

```
CREATE TABLE forum_votes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id),
    target_type VARCHAR(20) NOT NULL, -- 'post', 'comment'
    target_id UUID NOT NULL,
    vote_value INTEGER NOT NULL, -- 1 (upvote), -1 (downvote)
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(user_id, target_type, target_id),
    CONSTRAINT valid_target_type CHECK (target_type IN ('post', 'comment')),
    CONSTRAINT valid_vote_value CHECK (vote_value IN (-1, 1))
);

CREATE INDEX idx_forum_votes_user_id ON forum_votes(user_id);
CREATE INDEX idx_forum_votes_target ON forum_votes(target_type, target_id);
```

6.4 forum_tags

Predefined forum tags.

```
CREATE TABLE forum_tags (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    organization_id UUID REFERENCES organizations(id),
    name VARCHAR(100) NOT NULL,
```

```
    slug VARCHAR(100) NOT NULL,
    description TEXT,
    color VARCHAR(7), -- Hex color like '#FF5733'
    usage_count INTEGER DEFAULT 0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(organization_id, slug)
);

CREATE INDEX idx_forum_tags_organization_id ON forum_tags(organization_id);
CREATE INDEX idx_forum_tags_slug ON forum_tags(slug);
```

7. Notifications

7.1 notifications

User notifications.

```
CREATE TABLE notifications (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Notification details
    type VARCHAR(50) NOT NULL, -- 'submission_feedback', 'assessment_assigned', 'forum_repl
    title VARCHAR(255) NOT NULL,
    message TEXT NOT NULL,
    action_url TEXT,

    -- Status
    is_read BOOLEAN DEFAULT false,
    read_at TIMESTAMP WITH TIME ZONE,

    -- Delivery
    delivered_via TEXT[], -- ['in_app', 'email']
    email_sent_at TIMESTAMP WITH TIME ZONE,

    -- Metadata
    metadata JSONB DEFAULT '{}',

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_notifications_user_id ON notifications(user_id);
CREATE INDEX idx_notifications_is_read ON notifications(is_read);
CREATE INDEX idx_notifications_type ON notifications(type);
```

```sql
CREATE INDEX idx_notifications_created_at ON notifications(created_at DESC);
```

8. Activity Logs and Analytics

8.1 activity_logs

Audit trail of user actions.

```sql
CREATE TABLE activity_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    organization_id UUID REFERENCES organizations(id),

    -- Action details
    action_type VARCHAR(100) NOT NULL, -- 'login', 'submission_created', 'assessment_comple
    resource_type VARCHAR(50), -- 'submission', 'assessment', 'user'
    resource_id UUID,

    -- Context
    description TEXT,
    ip_address INET,
    user_agent TEXT,

    -- Metadata
    metadata JSONB DEFAULT '{}',

    -- Timestamp
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
) PARTITION BY RANGE (created_at);

CREATE INDEX idx_activity_logs_user_id ON activity_logs(user_id);
CREATE INDEX idx_activity_logs_action_type ON activity_logs(action_type);
CREATE INDEX idx_activity_logs_created_at ON activity_logs(created_at DESC);

-- Create partitions by month
CREATE TABLE activity_logs_2025_10 PARTITION OF activity_logs
    FOR VALUES FROM ('2025-10-01') TO ('2025-11-01');
-- Additional partitions created as needed
```

8.2 student_metrics

Aggregated student performance metrics (materialized view or scheduled calculation).

```sql
CREATE TABLE student_metrics (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    student_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

    -- Period
    metric_period VARCHAR(20) NOT NULL, -- 'all_time', 'current_month', 'current_week'
    period_start DATE,
    period_end DATE,

    -- STEM scores
    overall_stem_score INTEGER,
    programming_score INTEGER,
    engineering_score INTEGER,
    problem_solving_score INTEGER,
    documentation_score INTEGER,

    -- Activity stats
    total_submissions INTEGER DEFAULT 0,
    total_assessments INTEGER DEFAULT 0,
    assessments_passed INTEGER DEFAULT 0,
    forum_posts INTEGER DEFAULT 0,
    forum_comments INTEGER DEFAULT 0,

    -- Engagement
    days_active INTEGER DEFAULT 0,
    avg_submission_score DECIMAL(5,2),

    -- Trend
    score_trend VARCHAR(20), -- 'improving', 'stable', 'declining'

    -- Calculated at
    calculated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    UNIQUE(student_id, metric_period, period_start),
    CONSTRAINT valid_metric_period CHECK (metric_period IN ('all_time', 'current_month', 'cu
);

CREATE INDEX idx_student_metrics_student_id ON student_metrics(student_id);
CREATE INDEX idx_student_metrics_period ON student_metrics(metric_period);
```

9. Supporting Tables

9.1 ai_prompt_cache

Cache for AI prompts and responses to reduce costs.

```
CREATE TABLE ai_prompt_cache (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    prompt_hash VARCHAR(64) UNIQUE NOT NULL, -- SHA-256 hash of prompt
```

```
    prompt_template VARCHAR(100) NOT NULL,
    prompt_version VARCHAR(20) NOT NULL,
    model VARCHAR(50) NOT NULL,

    -- Cached response
    response_text TEXT NOT NULL,
    response_json JSONB,

    -- Metadata
    token_count INTEGER,
    cost DECIMAL(10,4),

    -- Cache management
    hit_count INTEGER DEFAULT 0,
    last_hit_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP WITH TIME ZONE DEFAULT (CURRENT_TIMESTAMP + INTERVAL '24 hours')
);

CREATE INDEX idx_ai_prompt_cache_hash ON ai_prompt_cache(prompt_hash);
CREATE INDEX idx_ai_prompt_cache_expires_at ON ai_prompt_cache(expires_at);
```

9.2 email_queue

Queue for outgoing emails.

```
CREATE TABLE email_queue (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    recipient_email VARCHAR(255) NOT NULL,
    recipient_name VARCHAR(255),
    user_id UUID REFERENCES users(id),

    -- Email details
    email_type VARCHAR(50) NOT NULL, -- 'welcome', 'password_reset', 'notification'
    subject VARCHAR(500) NOT NULL,
    body_html TEXT NOT NULL,
    body_text TEXT,

    -- SendGrid details
    template_id VARCHAR(100),
    template_data JSONB,

    -- Status
    status VARCHAR(20) DEFAULT 'pending', -- 'pending', 'sent', 'failed'
    attempts INTEGER DEFAULT 0,
```

```
    error_message TEXT,
    sent_at TIMESTAMP WITH TIME ZONE,

    -- Metadata
    metadata JSONB DEFAULT '{}',

    -- Audit
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT valid_status CHECK (status IN ('pending', 'sent', 'failed'))
);

CREATE INDEX idx_email_queue_status ON email_queue(status);
CREATE INDEX idx_email_queue_created_at ON email_queue(created_at);
```

10. Database Functions and Triggers

10.1 Update timestamp trigger

Automatically update `updated_at` on row changes.

```
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Apply to all tables with updated_at
CREATE TRIGGER update_users_updated_at
    BEFORE UPDATE ON users
    FOR EACH ROW
    EXECUTE FUNCTION update_updated_at_column();

-- Repeat for other tables...
```

10.2 Update vote counts

Update post/comment vote_count when votes change.

```
CREATE OR REPLACE FUNCTION update_vote_count()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        IF NEW.target_type = 'post' THEN
```

```
            UPDATE forum_posts SET vote_count = vote_count + NEW.vote_value WHERE id = NEW.t
        ELSIF NEW.target_type = 'comment' THEN
            UPDATE forum_comments SET vote_count = vote_count + NEW.vote_value WHERE id = NE
        END IF;
    ELSIF TG_OP = 'UPDATE' THEN
        IF NEW.target_type = 'post' THEN
            UPDATE forum_posts SET vote_count = vote_count - OLD.vote_value + NEW.vote_value
        ELSIF NEW.target_type = 'comment' THEN
            UPDATE forum_comments SET vote_count = vote_count - OLD.vote_value + NEW.vote_va
        END IF;
    ELSIF TG_OP = 'DELETE' THEN
        IF OLD.target_type = 'post' THEN
            UPDATE forum_posts SET vote_count = vote_count - OLD.vote_value WHERE id = OLD.t
        ELSIF OLD.target_type = 'comment' THEN
            UPDATE forum_comments SET vote_count = vote_count - OLD.vote_value WHERE id = OL
        END IF;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER forum_votes_update_count
    AFTER INSERT OR UPDATE OR DELETE ON forum_votes
    FOR EACH ROW
    EXECUTE FUNCTION update_vote_count();
```

10.3 Update comment count

Update post comment_count when comments change.

```
CREATE OR REPLACE FUNCTION update_comment_count()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        UPDATE forum_posts SET comment_count = comment_count + 1 WHERE id = NEW.post_id;
    ELSIF TG_OP = 'DELETE' THEN
        UPDATE forum_posts SET comment_count = comment_count - 1 WHERE id = OLD.post_id;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER forum_comments_update_count
    AFTER INSERT OR DELETE ON forum_comments
    FOR EACH ROW
```

```
    EXECUTE FUNCTION update_comment_count();
```

11. Materialized Views

11.1 Class performance summary

```
CREATE MATERIALIZED VIEW class_performance_summary AS
SELECT
    c.id AS class_id,
    c.name AS class_name,
    COUNT(DISTINCT cs.student_id) AS total_students,
    ROUND(AVG(sm.overall_stem_score), 2) AS avg_overall_score,
    ROUND(AVG(sm.programming_score), 2) AS avg_programming_score,
    ROUND(AVG(sm.engineering_score), 2) AS avg_engineering_score,
    COUNT(s.id) AS total_submissions,
    COUNT(DISTINCT s.student_id) FILTER (WHERE s.created_at >= CURRENT_DATE - INTERVAL '7 da
FROM classes c
LEFT JOIN class_students cs ON c.id = cs.class_id AND cs.status = 'active'
LEFT JOIN student_metrics sm ON cs.student_id = sm.student_id AND sm.metric_period = 'all_ti
LEFT JOIN submissions s ON cs.student_id = s.student_id
WHERE c.deleted_at IS NULL
GROUP BY c.id, c.name;

CREATE UNIQUE INDEX idx_class_performance_summary_class_id ON class_performance_summary(clas

-- Refresh daily
```

12. Database Maintenance

12.1 Scheduled Jobs (via pg_cron or external scheduler)

```
-- Daily: Delete expired sessions
DELETE FROM sessions WHERE expires_at < CURRENT_TIMESTAMP;

-- Daily: Delete expired password reset tokens
DELETE FROM password_reset_tokens WHERE expires_at < CURRENT_TIMESTAMP;

-- Daily: Delete old activity logs (keep 1 year)
DELETE FROM activity_logs WHERE created_at < CURRENT_TIMESTAMP - INTERVAL '1 year';

-- Daily: Refresh materialized views
REFRESH MATERIALIZED VIEW CONCURRENTLY class_performance_summary;

-- Weekly: Vacuum and analyze
VACUUM ANALYZE;

-- Weekly: Calculate student metrics
```

```
-- (Custom function to recalculate student_metrics)
```

Document Version: 1.0 **Last Updated**: 2025-10-18 **Next Review**: 2025-11-18