Users and their Authentication:

```
-- Users table (managed by Supabase Auth)
-- This table is automatically created by Supabase

-- User profiles
CREATE TABLE user_profiles (
  id UUID PRIMARY KEY REFERENCES auth.users(id),
  display_name TEXT NOT NULL,
  first_name TEXT,
  last_name TEXT,
  role TEXT NOT NULL CHECK (role IN ('student', 'teacher')),
  university TEXT,
  university_email TEXT,
  university_address TEXT,
  phone_number TEXT,
  semester TEXT,
  degree TEXT,
  designation TEXT,
  domain TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Add RLS policies for user_profiles
ALTER TABLE user_profiles ENABLE ROW LEVEL SECURITY;

-- Allow users to see their own profile
CREATE POLICY "Users can view their own profile"
  ON user_profiles FOR SELECT
  USING (auth.uid() = id);
```

Courses and Enrollment:

```
-- Categories for courses
CREATE TABLE course_categories (
  id SERIAL PRIMARY KEY,
  name TEXT UNIQUE NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Courses table
CREATE TABLE courses (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  title TEXT NOT NULL,
  description TEXT,
  instructor_id UUID REFERENCES user_profiles(id) NOT NULL,
  category_id INTEGER REFERENCES course_categories(id),
  semester TEXT,
  thumbnail_url TEXT,
  enrollment_key TEXT UNIQUE NOT NULL,
  is_published BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Course enrollment
CREATE TABLE course_enrollments (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  course_id UUID REFERENCES courses(id) ON DELETE CASCADE NOT NULL,
  student_id UUID REFERENCES user_profiles(id) ON DELETE CASCADE NOT NULL,
  enrolled_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  UNIQUE(course_id, student_id)
);

-- Then add this RLS policy to enforce the student role requirement
CREATE POLICY "Only student roles can be enrolled"
  ON course_enrollments FOR INSERT
  WITH CHECK (
    (SELECT role FROM user_profiles WHERE id = student_id) = 'student'
  );


-- Add RLS policies
ALTER TABLE courses ENABLE ROW LEVEL SECURITY;
ALTER TABLE course_enrollments ENABLE ROW LEVEL SECURITY;

-- Teachers can create, view and edit their own courses
CREATE POLICY "Teachers can manage their own courses"
  ON courses FOR ALL
  USING (auth.uid() = instructor_id);

-- All authenticated users can view published courses
CREATE POLICY "Users can view published courses"
  ON courses FOR SELECT
  USING (is_published = TRUE);

-- Students can enroll in courses
CREATE POLICY "Students can enroll in courses"
```

```
  ON course_enrollments FOR INSERT
  WITH CHECK (
    auth.uid() = student_id AND
    (SELECT role FROM user_profiles WHERE id = auth.uid()) = 'student'
  );

-- Students can view their enrollments
CREATE POLICY "Students can view their enrollments"
  ON course_enrollments FOR SELECT
  USING (auth.uid() = student_id);
```

Course Content

```
-- Lectures table
CREATE TABLE lectures (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  course_id UUID REFERENCES courses(id) ON DELETE CASCADE NOT NULL,
  title TEXT NOT NULL,
  display_order INTEGER NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Content types
CREATE TABLE content_types (
  id SERIAL PRIMARY KEY,
  name TEXT UNIQUE NOT NULL
);

-- Initial content types
INSERT INTO content_types (name) VALUES ('slides'), ('notes');

-- Lecture content
CREATE TABLE lecture_content (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  lecture_id UUID REFERENCES lectures(id) ON DELETE CASCADE NOT NULL,
  content_type_id INTEGER REFERENCES content_types(id) NOT NULL,
  file_name TEXT NOT NULL,
  file_url TEXT NOT NULL,
  file_size INTEGER,
  uploaded_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Add RLS policies
```

```
ALTER TABLE lectures ENABLE ROW LEVEL SECURITY;
ALTER TABLE lecture_content ENABLE ROW LEVEL SECURITY;

-- Teachers can create and manage lectures for their own courses
CREATE POLICY "Teachers can manage lectures for their courses"
  ON lectures FOR ALL
  USING (
    (SELECT instructor_id FROM courses WHERE id = course_id) = auth.uid()
  );

-- Teachers can manage content
CREATE POLICY "Teachers can manage lecture content"
  ON lecture_content FOR ALL
  USING (
    (SELECT instructor_id FROM courses
     JOIN lectures ON lectures.course_id = courses.id
     WHERE lectures.id = lecture_content.lecture_id) = auth.uid()
  );

-- Enrolled students can view content
CREATE POLICY "Enrolled students can view content"
  ON lecture_content FOR SELECT
  USING (
    EXISTS (
      SELECT 1 FROM course_enrollments
      JOIN lectures ON lectures.course_id = course_enrollments.course_id
      WHERE lecture_content.lecture_id = lectures.id
      AND course_enrollments.student_id = auth.uid()
    )
  );
```

Quizzes and Assignments

```
-- Difficulty levels
CREATE TABLE difficulty_levels (
  id SERIAL PRIMARY KEY,
  name TEXT UNIQUE NOT NULL
);

-- Insert initial difficulty levels
INSERT INTO difficulty_levels (name) VALUES ('easy'), ('medium'), ('hard');

-- Quizzes
CREATE TABLE quizzes (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  course_id UUID REFERENCES courses(id) ON DELETE CASCADE NOT NULL,
  lecture_id UUID REFERENCES lectures(id) ON DELETE SET NULL,
  title TEXT NOT NULL,
  key_topic TEXT,
  num_mcqs INTEGER,
  num_theory_questions INTEGER,
  num_numericals INTEGER,
  difficulty_id INTEGER REFERENCES difficulty_levels(id),
  num_versions INTEGER DEFAULT 1,
  generate_rubric BOOLEAN DEFAULT FALSE,
  additional_requirements TEXT,
  file_url TEXT,
  created_by UUID REFERENCES user_profiles(id) NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Assignments
CREATE TABLE assignments (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  course_id UUID REFERENCES courses(id) ON DELETE CASCADE NOT NULL,
  lecture_id UUID REFERENCES lectures(id) ON DELETE SET NULL,
  title TEXT NOT NULL,
  key_topic TEXT,
  num_numericals INTEGER,
  num_theory_questions INTEGER,
  total_marks INTEGER,
  difficulty_id INTEGER REFERENCES difficulty_levels(id),
  num_versions INTEGER DEFAULT 1,
  generate_rubric BOOLEAN DEFAULT FALSE,
  additional_requirements TEXT,
  file_url TEXT,
  created_by UUID REFERENCES user_profiles(id) NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Add RLS policies
ALTER TABLE quizzes ENABLE ROW LEVEL SECURITY;
ALTER TABLE assignments ENABLE ROW LEVEL SECURITY;

-- Policies for quizzes
CREATE POLICY "Teachers can manage quizzes for their courses"
  ON quizzes FOR ALL
  USING (
    (SELECT instructor_id FROM courses WHERE id = course_id) = auth.uid()
  );

-- Policies for assignments
CREATE POLICY "Teachers can manage assignments for their courses"
  ON assignments FOR ALL
  USING (
    (SELECT instructor_id FROM courses WHERE id = course_id) = auth.uid()
  );
```