DETAILED PROMPT FOR GITHUB COPILOT

Context You are assisting with the design and implementation of a PostgreSQL database using **Prisma ORM** for an AI-assisted quiz and evaluation system. The schema below is **final and authoritative**. Do not introduce additional tables or normalize further unless explicitly asked.

SYSTEM OVERVIEW

The system has **5 tables**, each with a clear, single responsibility:

1. **students** – stores student-related information
2. **teachers** – stores teacher-related information
3. **quizzes** – stores quiz metadata and questions created by teachers
4. **student_submissions** – stores what a student submitted for a quiz question
5. **submission_evaluations** – stores evaluation results for a submission

There are **no separate tables** for questions, quiz attempts, or answers.

HIGH-LEVEL ER RELATIONSHIPS (PK → FK)

- `teachers.id` → `quizzes.teacher_id`
- `students.id` → `student_submissions.student_id`
- `quizzes.id` → `student_submissions.quiz_id`
- `student_submissions.id` → `submission_evaluations.submission_id`

One submission has **at most one evaluation**.

BOX-STYLE RELATIONSHIP OVERVIEW

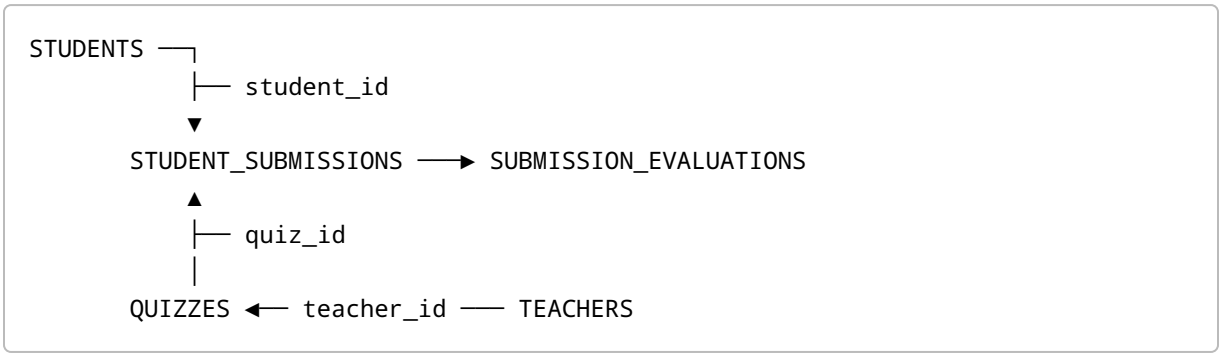

TABLE DEFINITIONS & RESPONSIBILITIES

1. students

Purpose: store student identity and academic information.

Core columns: - id (primary key) - name - email (unique) - roll_no (unique) - branch - semester

This table does **not** store quiz or submission data.

---

2. teachers

Purpose: store teacher identity information.

Core columns: - id (primary key) - name - email (unique) - branch

One teacher can create many quizzes.

---

3. quizzes

Purpose: store quiz metadata and all quiz questions.

Key design decision: **questions are stored as JSON**, not as a separate table.

Core columns: - id (primary key) - teacher_id (foreign key → teachers.id) - title - subject - course_code (e.g., CS458, EC778) - description (rules or instructions for students) - questions (JSON) - correct_answers (JSON) - start_time - end_time

Questions JSON Structure

Each question object may optionally include an image reference. Images are stored externally (CDN / cloud storage), and only references are kept.

Example:

```
[
  {
    "id": 1,
    "text": "Explain the OSI Model",
    "image": {
      "type": "url",
      "value": "https://cdn.app.com/images/osi.png"
    }
```

```
  },
  {
    "id": 2,
    "text": "What is normalization?",
    "image": null
  }
]
```

The `id` inside each question is a **logical identifier**, not a database foreign key.

---

4. student_submissions

Purpose: store what a student actually submitted for a quiz question.

Each row represents:

> one student answering one question from one quiz

Core columns: - id (primary key) - student_id (foreign key → students.id) - quiz_id (foreign key → quizzes.id) - question_id (logical reference to quizzes.questions JSON) - audio_codec - audio_path (reference only, no binary storage) - transcribed_answer - submitted_at

This table replaces traditional quiz-attempt or answer tables.

---

5. submission_evaluations

Purpose: store evaluation results for a student submission.

Core columns: - id (primary key) - submission_id (foreign key → student_submissions.id, unique) - actual_answer - similarity_score - marks_awarded - evaluated_at

Important constraints: - No AI feedback text is stored - One-to-one relationship with student_submissions

---

DESIGN CONSTRAINTS & INTENTIONAL DECISIONS

- No image or audio binaries are stored in the database
- Only paths or URLs are stored for media
- JSON is intentionally used to reduce table count and joins
- Strict normalization is relaxed for simplicity and faster development
- Schema is designed for MVP and can be normalized later if required

---

WHAT YOU (COPILOT) SHOULD DO

- Generate Prisma models that exactly follow this schema
- Generate queries, migrations, and constraints based on this design
- Do NOT introduce extra tables (e.g., questions, attempts, answers)
- Assume PostgreSQL as the database backend

This schema should be treated as **final and authoritative**.