□ Proof of Work: DBMS Mini Project – Student Attendance System

Submitted by: *Samar Joshi*
Branch & Section: *Computer Engineering, A*

□ Database Design Summary

My project fully implements a **relational database model** using **PostgreSQL**, and handles real-world complexities involved in managing a college-level **Student Attendance System**. The schema has been designed using **Prisma ORM** for structured and optimized queries.

□ Core Relationships & Integrity Constraints

□ 1. User Roles & Role-based Access Control

- One central `User` model connected via **1:1 relationships** to:
  - `Admin`
  - `Teacher`
  - `Student`
- Role is enforced via a custom enum (`Role`), ensuring access control is enforced through **user type segregation**.

□ 2. Student ↔ Enrollment ↔ Course Relationship

- A `Student` can enroll in **multiple courses** via the `Enrollment` table.
- Each enrollment captures:
  - **Semester**
  - **Academic Year**
  - **Section**
  - **Active status**
- **Integrity constraint:**

```
@@unique([studentId, courseId, semester, academicYear])
```

  This ensures a student cannot be enrolled in the same course more than once for the same semester and year.

□ 3. Teacher Assignment to Courses (TeachingAssignment)

- A `Teacher` can be assigned to **multiple courses**, branches, and sections.
- Each teaching assignment tracks:
  - **Course**
  - **Branch**
  - **Semester**
  - **Section** (Enum)
  - **Academic Year**
- **Constraint:**

```
@@unique([teacherId, courseId, branchId, semester, academicYear])
```

Ensures **no duplicate assignments** for the same course-teacher-branch-semester combination.

- **Demonstrates handling of multiple teachers for the same course in different sections of the same branch.**

4. **Section-wise Handling**

- `Section` is modeled as an enum (e.g., A, B), attached at both `Student` and `TeachingAssignment` level.
- This allows section-level differentiation within a branch and ensures **separate tracking of attendance and assignments**.

5. **Attendance Tracking with Session Mapping**

- `Session` is linked to `TeachingAssignment` to track each instance of a lecture.
- `Attendance` connects:
  - `Student`
  - `Session`
  - `Enrollment`
- **Constraint:**

```
@@unique([sessionId, studentId])
```

Prevents duplicate attendance entries for the same session-student pair.

6. **Cascade Deletion & Referential Integrity**

- Foreign key relationships with `onDelete: Cascade` in:
  - `User → Student/Teacher/Admin`
  - `Student → Enrollment`
  - `Session → Attendance`
- Ensures **automatic cleanup** of dependent records and maintains referential integrity.

WhatsApp Bot Integration

- Teachers can now:
  - Log in via WhatsApp
  - View course assignments
  - Mark attendance
  - Track sessions
- This integration demonstrates **practical implementation of DBMS logic over a messaging interface**, boosting usability and real-world alignment.

DBMS Concepts Implemented

- **Entity-Relationship Modeling**
- **One-to-One, One-to-Many, and Many-to-Many relationships**
- **Normalization (up to 3NF)**
- **Cascade Deletes**
- **Constraints (`@unique`, `@relation`, enums)**
- **Composite Unique Constraints**
- **Real-world modeling** of branches, sections, assignments, and attendance
- **Role-based access control and privilege hierarchy**

Final Note & Humble Request

Sir, I've genuinely put my best effort into translating the theoretical concepts of DBMS into a practical, scalable, and modular project. Every relationship and constraint in the schema was designed thoughtfully, keeping real-world college operations in mind.

Currently, I am at **65 marks**, and with just a **grace of 5 marks**, my grade will improve. I humbly request you to kindly consider this submission and the hard work behind it. A small grace would make a significant difference in my result, and I would be truly grateful for your kindness