Backend Development Assignment: Student Course Enrollment System

Objective

Design and implement a backend system for a student course enrollment platform using a database and a programming language of your choice. The system manages colleges, students, courses, and course timetables, ensuring students can select courses without timetable conflicts.

Requirements

1. Database Design

- **Description**: Design a database schema to support the following:
  - A list of colleges onboarded onto the platform.
  - Each college has a list of enrolled students for the semester.
  - Each college offers various courses (e.g., CS101, MA204, AP105).
  - Each course has a timetable (e.g., Monday 9AM–10AM, Tuesday 10AM–11AM, 3PM–6PM).
  - Students can select courses from their college, ensuring no timetable clashes.
- **Entities and Relationships**:
  - Colleges, with attributes like college ID and name.
  - Students, with attributes like student ID, name, and their associated college.
  - Courses, with attributes like course ID, code (e.g., CS101), and associated college.
  - Timetables, defining the schedule for each course (day, start time, end time).
  - Student course selections, linking students to their chosen courses.
- **Constraints**:
  - Ensure students and courses are associated with the same college.
  - Prevent students from selecting courses with clashing timetables.

2. Save Operation for Student Course Selection

- **Description**: Implement a function in a programming language of your choice to save a student's course selection to the database.
- **Function Requirements**:
  - Input: Student ID, a list of course IDs to enroll in.

- Validation:
  - Verify the student and courses exist and belong to the same college.
  - Check that the selected courses' timetables do not clash (i.e., no overlapping time slots on the same day).
- Output: Save the valid course selections to the database or return an error if validation fails (e.g., invalid student/course, timetable clash).
- Handle edge cases (e.g., empty course list, non-existent IDs).

3. Bonus: Database Constraints and Admin Functionality

- **Database Constraints**:
  - Implement database-level constraints to prevent invalid data, such as:
    - Ensuring students and their selected courses belong to the same college.
    - Preventing students from enrolling in courses with clashing timetables.
  - Example: Use foreign keys, unique constraints, or triggers to enforce these rules.
- **Admin Functionality**:
  - Allow administrators to add, edit, or remove a course's timetable.
  - Ensure timetable changes comply with the constraint that students and courses must belong to the same college.
  - Prevent timetable updates that would cause clashes for already enrolled students.

Deliverables

- **Database Schema**:
  - Submit a SQL file (e.g., `schema.sql`) or a clear description (e.g., in markdown or a diagram) defining tables, columns, data types, and relationships.
  - Include any constraints (e.g., foreign keys, checks) for the bonus section if implemented.
- **Source Code**:
  - Submit a file (e.g., `enrollment.py`, `enrollment.js`, etc.) containing the course selection save function.
  - Include necessary database connection code if applicable.
  - Package in a ZIP or GitHub repository.