Problem Statement: University Course and Student Management

Scenario:

You need to build a system component to manage students, courses, and enrollments at a university. Students can enroll in multiple courses, and courses can have multiple students. Additionally, each course belongs to a specific department. You will use SQLModel to model this data and perform operations.

Objective:

Implement SQLModel classes for Departments, Students, Courses, and Enrollments. Perform CRUD operations on Departments, Students, and Courses. Implement functionality to enroll/unenroll students in courses and retrieve relationship data.

Tasks:

1. **Define Models:**
    - **Department Model:**
        - id: Integer, primary key, optional.
        - name: String, required, unique.
        - building: String, optional.
        - courses: List of Course objects (relationship). Use Relationship(back_populates=...).
    - **Student Model:**
        - id: Integer, primary key, optional.
        - first_name: String, required.
        - last_name: String, required.
        - email: String, required, unique.
        - courses: List of Course objects (relationship via link table). Use Relationship(back_populates=..., link_model=...).
    - **Course Model:**
        - id: Integer, primary key, optional.
        - title: String, required, indexed.
        - code: String, required, unique.
        - department_id: Integer, foreign key referencing department.id, optional.
        - department: Optional Department object (relationship). Use Relationship(back_populates=...).
        - students: List of Student objects (relationship via link table). Use Relationship(back_populates=..., link_model=...).

talenciaglobal

- **StudentCourseLink Model (Link Table):**
    - student_id: Integer, foreign key referencing student.id, primary key.
    - course_id: Integer, foreign key referencing course.id, primary key.
    - enrollment_date: Optional datetime (use datetime from Python's datetime module). *This is a relationship attribute.*
    - grade: Optional String (e.g., "A", "B+", "In Progress"). *Another relationship attribute.*

2. **Database Setup:**
    - Create a SQLAlchemy engine connected to a SQLite database university.db.
    - Create all the necessary tables based on your models.

3. **CRUD Operations (Departments, Students, Courses):**
    - Write functions to create_department, create_student, create_course.
    - Write functions to get_department_by_name, get_student_by_email, get_course_by_code.
    - Write functions to list all entities (e.g., list_all_students).
    - Write a function to update a student's email (update_student_email).
    - Write a function to delete a course by its code (delete_course).
    - *Remember:* When creating a course, associate it with an existing department by setting department_id or the department object.

4. **Manage Enrollments (Many-to-Many Relationship):**
    - Write a function enroll_student(student_id: int, course_id: int, enrollment_date: Optional[datetime] = None):
        - Creates an entry in the StudentCourseLink table.
        - Handle potential errors (e.g., student or course not found, student already enrolled).
    - Write a function get_courses_for_student(student_id: int):
        - Retrieves a student.
        - Returns the list of courses they are enrolled in by accessing the student.courses relationship attribute. Print the course titles.
    - Write a function get_students_in_course(course_id: int):
        - Retrieves a course.
        - Returns the list of students enrolled by accessing the course.students relationship attribute. Print the student names.
    - Write a function set_enrollment_grade(student_id: int, course_id: int, grade: str):
        - Finds the specific StudentCourseLink entry for the given student and course.
        - Updates the grade attribute on that link table entry.
        - Commits the change.
    - Write a function unenroll_student(student_id: int, course_id: int):
        - Finds and deletes the corresponding entry from the StudentCourseLink table.

5. **Demonstrate Usage:**
    - In a if __name__ == "__main__": block:
        - Setup the database and tables.
        - Create a couple of departments (e.g., "Computer Science", "Mathematics").
        - Create a few courses, associating them with departments.
        - Create several students.
        - Enroll students in various courses (demonstrating Many-to-Many).
        - List the courses for a specific student.
        - List the students in a specific course.
        - Set a grade for one of the enrollments.
        - Unenroll a student from a course.
        - List the students in that course again to verify the unenrollment.
        - Perform basic CRUD operations (update a student, delete a course) and verify.