Short design explanation document :

1. Structural Overview: MVC

The application is divided into three main layers to separate concerns:

- **Models:** Define the data structure (Course and Student).
- **Views:** HTML templates (Jinja2) for the user interface.
- **Controllers:** Handle user requests and coordinate between Services and Views.

---

2. Design Pattern Implementations

A. Singleton Pattern (Resource Management)

Location: config/database.py The **Singleton** pattern is applied to the database connection. By using the __new__ method, we ensure that only one instance of the connection exists throughout the application's lifecycle.

- **Benefit:** Prevents multiple connections from slowing down the system and avoids "Database is locked" errors in SQLite.

B. Repository Pattern (Data Abstraction)

Location: repositories/ folder We decoupled the data access logic from the Models. The **Repositories** contain all the raw SQL queries.

- **Benefit:** If the database type changes (e.g., from SQLite to PostgreSQL), only the Repository files need to be modified; the Models and Services remain untouched.

C. Factory Pattern (Object Creation)

Location: factories/repoFactory.py The **Factory** pattern provides a centralized interface for creating objects. Instead of the Service layer manually importing and instantiating different repositories, it calls RepositoryFactory.get_repository('type').

- **Benefit:** Simplifies the Service layer and makes it easier to add new entities (like "Instructors" or "Grades") in the future.

---

3. System Flow

1. **Request:** User accesses /courses.
2. **Controller:** courseController receives the request.
3. **Service:** Calls courseService, which asks the **Factory** for a **Repository**.
4. **Repository:** Uses the **Singleton** connection to fetch data and maps it to a **Model**.
5. **Response:** The Controller sends the data to the **View** for rendering.