🎯 Objective:

Your task is to **design, develop, and deploy a modular web application** using the **MVC pattern** for the UI layer and **Microservices architecture** for backend APIs. You must ensure that the application is testable, modular, scalable, and integrated into a **CI/CD pipeline using Jenkins**.

🏗 Project Scope:

You can select a domain of your choice (e.g., Online Shopping, Healthcare System, Student Management System, Booking System, Inventory Management, etc.) but your solution must fulfill the following **technical** and **functional** requirements.

🛠 Technical Requirements:

1. Architecture

- The **UI** must follow the **MVC pattern** using a framework of your choice (e.g., ASP.NET Core MVC, Django, Spring MVC).
- The **backend must be implemented using Microservices**:
  - Split functionalities into **at least two independent services** (e.g., User Service, Product Service, Order Service).
  - Services should communicate via **HTTP APIs** or **message queues** (if applicable).

2. Frontend/UI

- UI should:
  - Be interactive and responsive.
  - Consume microservices via APIs (not directly accessing databases).
  - Have at least 3 functional screens (e.g., Create, Read, Update operations).

3. Microservices APIs

- Implement RESTful APIs with full **CRUD** operations.
- Each microservice must:
  - Run independently.
  - Connect to its own **dedicated database**.
  - Be version-controlled and documented.

4. API Testing

- Use **Postman** to create and validate API requests and responses.
  - OR
- Automate API tests using **Playwright** with proper assertions.
- Include:
  - Test collections (Postman) OR test scripts (Playwright).
  - At least 5 functional API test cases per service.

5. CI/CD Integration

- Use **Jenkins** to create a pipeline that:

- Builds the application or services.
- Runs automated tests (unit or API tests).
- Optionally, packages or deploys the solution.

📦 Deliverables:

1. ✅ Working microservices-based web application (UI + APIs).
2. ✅ Source code hosted on GitHub/GitLab (each microservice in its own folder or repo).
3. ✅ API testing:
    - Postman collection (.json) OR
    - Playwright test scripts (.js/.ts/.cs)
4. ✅ Jenkinsfile or Jenkins pipeline configuration with documentation.
5. ✅ A 5–10 minute final demo video or presentation.
6. ✅ README file with:
    - System architecture diagram.
    - Setup/run instructions.
    - API documentation.