Problem Statement: Online Learning Portal (Mini Udemy Clone)

1. Objective

Design and develop a backend system for an Online Learning Portal that allows users to register, browse courses, enroll, consume course content, and submit assignments. The system should support microservices architecture, use SQL (MySQL/PostgreSQL) for structured data, apply Spring Boot with JPA and WebFlux, and integrate JWT authentication, pagination, and DevOps basics.

2. Functional Requirements

- User Management: Register, login (JWT), update profile, admin manages courses.
- Course Management: CRUD courses, categories, tags, pagination.
- Enrollment & Progress: Enroll, track progress, list enrolled courses.
- Content Delivery: Videos, PDFs, pagination for large lists.
- Assignments & Submissions: Admin creates assignments, students submit solutions, grading system.
- Notifications (Optional): Email notifications for enrollments and deadlines.

3. Non-Functional Requirements

Authentication & Security (JWT, Spring Security), Scalability with microservices, SQL database (with possible NoSQL for content), pagination, CI/CD with GitHub Actions or Jenkins, Dockerization, testing with JUnit/Mockito.

4. System Design

Microservices: User, Course, Enrollment, Content, Assignment, Notification (optional).

Database ERD (SQL):

- users (id, name, email, password, role, created_at, updated_at)
- courses (id, title, description, category, tags, created_by, created_at)
- enrollments (id, user_id, course_id, progress, enrolled_at)
- contents (id, course_id, type, url, display_order)
- assignments (id, course_id, title, description, due_date)
- submissions (id, assignment_id, user_id, file_url, submitted_at, grade)

5. APIs

- User Service: POST /auth/signup, POST /auth/login, GET /users/{id}
- Course Service: GET /courses?page=1&size;=10, GET /courses/{id}, POST /courses (ADMIN)
- Enrollment Service: POST /enrollments, GET /enrollments/user/{id}
- Assignment Service: POST /assignments (ADMIN), POST /submissions (STUDENT), GET /submissions/user/{id}

6. Implementation Steps (Roadmap)

- Setup: Create GitHub repo, push Spring Boot project.
- User Service: Entity, repository, service, controller, JWT auth.
- Course Service: Entity + CRUD APIs, pagination.
- Enrollment Service: Manage enrollments, track progress.
- Content & Assignments: Add APIs for content and submissions.
- Testing: JUnit + Mockito.
- DevOps: Dockerize, setup GitHub Actions/Jenkins, deploy to AWS EC2.

7. Deliverables

- Source code in GitHub (with README).
- ER diagram (dbdiagram.io export).
- Postman collection of APIs.
- Docker setup for local deployment.
- CI/CD pipeline configuration.

8. Resume Highlights

- Built microservices-based backend using Spring Boot, JPA, WebFlux.
- Designed relational schema & ERD with SQL.
- Implemented JWT authentication, role-based access, pagination.
- Integrated CI/CD pipeline with GitHub Actions/Jenkins & Docker.
- Applied OOP principles, clean architecture, testing with JUnit/Mockito.