High-Level Design (HLD) for Online Learning Portal

1. Project Overview

An online learning portal where:

- Users can browse courses, enroll, watch lectures, submit assignments, and track progress.
- Admins can create, update, and delete courses.
- Supports pagination, reactive APIs, and secure authentication.
- Built with Spring Boot, Java, Spring Reactive, and MongoDB/MySQL.
- Microservices-based architecture.
- CI/CD pipelines (Jenkins/GitHub Actions) and basic DevOps practices.

2. Functional Modules

Module	Description
User Management	Signup/Login, JWT-based authentication, OAuth support, profile management.
Course Management	CRUD operations on courses, upload content (video/PDF), categories/tags.
Enrollment & Progress	Users can enroll in courses, track progress, submit assignments/quizzes.
Content Delivery	Streaming videos, PDF downloads, pagination for course listings.
Admin Panel	Manage users, courses, categories, view analytics.
Notification Module	Email/SMS notifications for course updates, assignments, reminders.

3. Microservices Architecture

1. User Service - authentication, authorization, profile management.
2. Course Service - CRUD for courses, categories, tags.
3. Enrollment & Progress Service - track enrollments, progress, submissions.
4. Content Service - streaming videos, PDFs, pagination.
5. Notification Service - email/SMS notifications.

4. Technology Stack

Layer	Technology
Backend	Java 17, Spring Boot, Spring Reactive (WebFlux)
Database	MongoDB, MySQL/PostgreSQL
Authentication	JWT, Spring Security, OAuth2
API Design	REST + Reactive endpoints
DevOps	Git/GitHub, Jenkins/GitHub Actions, Docker, AWS/EC2
Other	Pagination, DTOs, Logging (SLF4J), Exception Handling

5. Database Design (High-Level ERD)

Entities:

- User: id, name, email, password, role, created_at
- Course: id, title, description, category, tags, created_by, created_at
- Enrollment: id, user_id, course_id, progress, enrolled_at
- Content: id, course_id, type, url, order
- Assignment: id, course_id, title, description, due_date
- Submission: id, assignment_id, user_id, submitted_file_url, submitted_at, grade

6. API Endpoints

User Service:

- POST /api/auth/signup
- POST /api/auth/login
- GET /api/users/{id}

Course Service:

- GET /api/courses?page=1&size=10
- POST /api/courses
- GET /api/courses/{id}

Enrollment Service:

- POST /api/enrollments
- GET /api/enrollments/user/{userId}

7. System Design Considerations

- Scalability: Microservices
- Data Storage: MongoDB for content, SQL for structured data
- Caching: Redis
- Reactive APIs: Spring WebFlux
- Security: JWT, Role-based access control, OAuth2
- Pagination & Sorting: Standard REST query params
- Monitoring & Logging: Spring Boot Actuator, ELK (optional)

8. CI/CD & DevOps
- Version Control: Git/GitHub
- Build & Test: Maven/Gradle
- CI/CD: Jenkins/GitHub Actions, automated tests, Docker build & push, deploy to EC2/AWS
- Environment Management: Properties/YAML per environment

9. Resume Highlights
- Microservices-based architecture
- Reactive programming with Spring WebFlux
- JWT authentication and OAuth integration
- Database handling (SQL + NoSQL)
- Pagination, filtering, sorting in APIs
- CI/CD pipelines
- OOP principles (SOLID, DTOs, service-layer abstraction)
- Git/GitHub workflow
- Basic DevOps awareness (Docker, Jenkins, AWS deployment)