2025 Spring

SWER313 - SERVICE ORIENTED ARCHITECTURE

SWER354 - ADVANCED WEB TECHNOLOGIES

Course Project

Learning Management System (LMS) Development

Project Overview

In this project, you will develop a **Learning Management System (LMS)** using two different architectural approaches:

- **Step 1:** Build the LMS as a **monolithic** application.
- **Step 2:** Develop a **frontend** (React) that connects to the backend.
- **Step 3:** Refactor and convert it into a **microservices-based** system.

This project will help you understand the **challenges and advantages** of each approach, enabling you to experience real-world **service-oriented architecture** principles.

Step 1: Monolithic LMS Development

Deadline: 19/03/2025

Objective:

Develop a **single, monolithic application** that implements the core functionalities of an LMS.

Requirements

Your monolithic LMS should include the following **core modules** within a single codebase:

1. **User Management**:
   - Register/login users and assign roles (**Admin, Instructor, Student**).
   - **Implement JWT/OAuth2 authentication** for secure access.
   - Users should be able to update their profiles.
2. **Course Management**:
   - Create, update, delete, and list courses.
   - Assign instructors to courses.
3. **Enrollment System**:
   - Students can enroll/unenroll in courses.
   - Track student progress and course completion.

4. **Content Management**:
    - Instructors can upload lectures (**PDFs, videos, quizzes**).
    - Students can view/download content.
5. **Assessment Module**:
    - Create quizzes and assignments.
    - Auto-grade quizzes and store scores.
6. **Notification System**:
    - Send **email/SMS notifications** when a course is updated or an assignment is due.

Technical Constraints

- Use **Spring Boot (Java)** for the backend.
- Use **PostgreSQL/MySQL** as the relational database.
- Implement **JWT/OAuth2 authentication** for secure API access.
- The LMS should run as a **single deployable unit**.

Submission Requirements

- **GitHub Repository** with a detailed `README.md`.
- **Postman Collection** or API documentation
- **Database schema** (ERD Diagram).

Step 2: Frontend Development

Deadline:

Objective:

Develop a **frontend application** that interacts with the monolithic backend.

Frontend Requirements

Your frontend should:

- **Consume REST APIs** from the monolithic backend.
- Provide a **user-friendly UI** for:
    - User login/signup.
    - Course browsing and management.
    - Enrollment and progress tracking.
    - Quiz and assignment submission.
- Implement **JWT authentication** to access protected resources.

Technical Constraints

- **Frontend Framework:** React.js.
- **UI Components:** Material UI, Bootstrap, or Tailwind CSS.
- **State Management:** React Context API or Redux.
- **Security:** Store and manage JWT tokens securely.

Submission Requirements

- **GitHub Repository** with a working frontend.
- **Demo** showing the UI.

Step 3: Convert Monolithic LMS to Microservices

Deadline: 20/05/2025

Objective:

Refactor your monolithic LMS into a **microservices-based** system using **Service-Oriented Architecture (SOA)** principles.

Microservices Breakdown

Your new system should **split functionalities into separate microservices**, each with its own database:

Microservice	Responsibilities
User Service	Authentication, roles (Admin, Instructor, Student)
Course Service	Manage courses (create, update, delete, list)
Enrollment Service	Handle student enrollments and progress tracking
Content Service	Manage course materials (PDFs, videos)
Assessment Service	Create and grade quizzes, track performance
Notification Service	Send emails and push notifications

Technical Requirements

- **Convert monolithic APIs into independent microservices**.
- Implement **inter-service communication** using **REST APIs**.
- Use **RabbitMQ** for messaging between services.
- Implement **API Gateway** to route all requests.
- Each microservice should have **its own database**.
- **Maintain JWT/OAuth2 authentication** across microservices.
- Deploy the microservices using **Docker** to **AWS** (Optional for extra credit).

Submission Requirements

- **Separate GitHub repositories** for each microservice.
- **Docker Compose YAML** for deployment.
- **Postman Collection** or API documentation.
