E-learning Application

Overview:

In this project, students will be required to build an E-learning application using microservices architecture, MongoDB, Spring Boot, Kafka/RabbitMQ, and Docker. The application will consist of 10 microservices that will communicate with each other via Kafka or RabbitMQ queues. Each team will be assigned two microservices and an API Gateway. The application will be built using Java and the Spring Boot framework. Unit tests will be implemented to test the producers and consumers of each component in the microservices.

Objectives:

- Build a scalable and robust E-learning application using a microservices architecture.
- Implement secure user authentication and authorization using Spring Security.
- Work with Kafka or RabbitMQ queues to exchange messages between microservices.
- Containerize the application using Docker for easy deployment.
- Implement MongoDB as a NoSQL database to store and retrieve application data.
- Develop and implement unit tests to ensure the proper functioning of each component in the microservices.
- Work collaboratively in a team environment and practice agile project management techniques.

Scope:

The project will be divided into the following phases:

1. Planning and Requirements Gathering
2. Microservices Design and Implementation
3. API Gateway Implementation
4. Kafka or RabbitMQ Integration
5. MongoDB Database Design and Implementation
6. Dockerization
7. Unit Testing

Microservices Description:

Profile - Manages user-profiles and account settings.

- Consume: File uploaded, File deleted
- Produce: User created, User updated, User deleted

Post - Allows users to create, edit, and view posts related to course discussions.

- Consume: Comment created, Comment updated, Comment deleted
- Produce: Post created, Post updated, Post deleted

Comment - Enables users to add, edit, and view comments on posts or material.

- Produce: Comment created, Comment updated, Comment deleted

Order - Manages orders for courses, including enrollment and cancellation.

- Consume: Material created, Material updated, Material deleted
- Produce: Order created

Coin - Handles the virtual coin balance of users, transactions and coin packages.

- Consume: Post created, Payment processed, Order created
- Produce: Coins purchased, Coins used, Coins added, package added, package updated, package deleted

Payment - Handles payment gateway integration for purchases.

- Consume: package added, package updated, package deleted
- Produce: Payment processed, Payment failed, Subscription created, Subscription updated, Subscription canceled

File - Stores and retrieves files related to course materials, images or profile image .

- Consume: User deleted, Material deleted
- Produce: File uploaded, File deleted

Material - Manages course materials, including videos, articles, and documents.

- Consume: Comment created, Comment updated, Comment deleted, File uploaded, File deleted
- Produce: Material created, Material updated, Material deleted

Curriculum - Manages course curriculums, including course modules and lessons.

- Produce: Curriculum created, Curriculum updated, Curriculum deleted

Question - Handles user-generated questions related to course content.

- Consume: Curriculum created, Curriculum updated, Curriculum deleted, Material created, Material updated, Material deleted
- Produce: Question created, Question updated, Question deleted

Unit Testing:

To ensure the proper functioning of each component in the microservices, unit tests will be developed and implemented using Junit. Each microservice will have its own set of unit tests for the producers and consumers of messages on the Kafka or RabbitMQ queues. Additionally, integration tests will be developed to test the communication between microservices. The tests will cover both positive and negative scenarios to ensure the application functions as expected.

Libraries:

The project will make use of popular libraries such as Mapstruct and Lombok to reduce boilerplate code and increase code readability. If needed, Querydsl can also be used for query building and manipulation.

Team Composition:

Each team will consist of 3 students

Deliverables:

- A working E-learning application with your microservices and API Gateway implemented.
- A Docker container with the application deployed.
- Documentation outlining the design and implementation of the application.
- A presentation outlining the project and its implementation.

Grade Schema:

1. Functionality.
2. Microservices Architecture.
3. Database Design and Implementation.
4. Dockerization.The application meets all functional requirements, including creating, editing, and viewing posts, comments, and course materials, managing user profiles, processing payments, managing orders and enrollments, and handling user-generated questions related to course content.
5. Testing.

Team	Microservices
Team 1	Post, Comment
Team 2	Coin
Team 3	Profile, Curriculum
Team 4	Question, File
Team 5	Material
Team 6	Payment, Order