Lab Course 2

Course Overview

In Lab Course 2, students will engage in a comprehensive software development project encompassing both theoretical knowledge and practical skills. This project aims to simulate a real-world software development environment where students will develop a full-stack application. The application will include a frontend, a backend, and will integrate both SQL and NoSQL databases. Students are required to manage their codebase using Git and coordinate their tasks using a project management tool.

Project Mandatory Requirements

1. **Technology Stack:**
   - **Frontend Framework**: Students must choose and implement **one** frontend framework (e.g., React, Angular, Vue.js).
   - **Backend Framework**: Students must choose and implement **one** backend framework (e.g., ASP.NET, Spring Boot, Express.js, Django, Flask).
   - **SQL Database**: Students are required to integrate a SQL database into their application (e.g., MySQL, PostgreSQL).
   - **NoSQL Database**: Students are required to integrate a NoSQL database into their application (e.g., MongoDB, Cassandra, Firebase Firestore).
2. **Version Control**

   The use of Git for version control is mandatory. Students should maintain a clean and organized repository, with meaningful commit messages.

3. **Project Management Tool**

   Students must use a task management tool to plan, track, and collaborate on various aspects of the project (e.g., Trello, Jira).

Additional Requirements

Enhance your project by implementing additional features. The number of features implemented will affect the maximum grade achievable.

Choose from the following features:

1. **Authentication and Authorization**: Implement secure authentication and authorization using Refresh and Access tokens to manage user sessions and access controls.
2. **Real-Time Communication using WebSockets**: Develop a feature that requires real-time communication between the client and server using WebSockets.
3. **Real-Time Communication using WebRTC**: Integrate real-time, peer-to-peer communication capabilities in your application using WebRTC.
4. **Real-Time Notifications**: Implement a system to push notifications to users in real-time for various events, such as new messages, updates, or alerts. This can be achieved using technologies like WebSockets or service workers for web applications.
5. **Advanced Search Functionality**: Incorporate advanced search capabilities with filters, sort options, and full-text search to help users find specific information quickly and efficiently. Using technologies like Elasticsearch can improve search performance and accuracy.
6. **Content Management System (CMS)**: Include a CMS to allow non-technical users to update content on the website without needing to understand the underlying code. This can empower content creators and reduce the workload on developers.
7. **Microservices Architecture**: Design your application using a microservices architecture, including an API Gateway, Docker(to containerize microservices), load balancer, and proper communication between services.

8. **Hexagonal Architecture Implementation**: Design your application following the hexagonal architecture pattern to ensure a clean separation between the core business logic and external interfaces. This approach involves structuring the application so that the core logic communicates with databases, services, and user interfaces through ports and adapters.
9. **Online Payment Integration**: Incorporate an online payment solution to enable financial transactions within your application.
10. **Data Exporting and Importing**: Enable users to export data from reports into various formats (e.g., CSV, Excel, JSON) for further analysis or import external data to be included in reports. This feature enhances the flexibility and usability of the report generation system.
11. **Dynamic Report Generation**: Allow users to generate customizable reports based on selected criteria, such as date ranges, specific data fields, or based on user activity and feedback within the application. This feature would require implementing a flexible backend logic to query the database dynamically based on user inputs and a frontend interface for specifying report criteria.
12. **Machine Learning Integration**: Implement a machine learning model to provide intelligent features such as personalized recommendations, predictive text, or image recognition within your application. This could involve using pre-trained models or developing custom models tailored to your application's needs.

Team Composition Requirements:

1. **Group Size**: Each project team must consist of exactly **5 students**. No more, no less. This requirement is designed to ensure that each team member can substantially contribute to the project while learning to navigate the complexities of teamwork and project management.
2. **Collaboration**: Effective collaboration within each group is crucial. Teams are encouraged to leverage the diverse skills and knowledge of all members, fostering an environment of peer learning and shared responsibility.
3. **Roles and Responsibilities**: Teams should define clear roles and responsibilities for each member, aligning with the project's needs and each member's strengths. This might include specific focuses on frontend development, backend development, database management, documentation, and project management.

Students Evaluation Breakdown

1. **Project Mandatory Requirements (60%)**
    a. Frontend Framework Implementation: **10%**
    b. Backend Framework Implementation: **10%**
    c. SQL Database Integration: **10%**
    d. NoSQL Database Integration: **10%**
    e. Version Control with Git (includes commit history): **10%**
    f. Use of Project Management Tool (evidence of task tracking, collaboration, and project planning): **10%**
2. **Additional Features Implementation (30%)**
    a. First feature implementation: **10%**
    b. Second feature implementation: **10%**
    c. Third feature implementation: **10%**
3. **Documentation (10%)**