Project Requirements: SkillStream Web Application

Introduction

The SkillStream Web Application is a platform designed to provide a seamless learning experience to students. The platform is intended to be user-friendly and easy to navigate. The goal is to create a comprehensive learning platform that includes a variety of courses for students to choose from.

Functional Requirements:

1. User Authentication:

The system should allow students to sign up and log in with their email address and password. The admin should be able to log in using a separate admin account. The system should have a secure authentication mechanism to prevent unauthorized access.

2. Course Management
   The system should provide a list of available courses for students to browse and search. The admin should be able to add new courses, edit existing courses, and delete courses that are no longer offered. Students should be able to enroll in courses they are interested in, either by clicking an "Enroll" button
3. Course Ratings and Reviews:
   Students should be able to rate and review courses they have taken, which helps other students decide which courses to enroll in. The system should display an average rating and the number of reviews for each course, and allow students to write their own reviews.
4. Security:
   The system should be secure and protect user data, including using HTTPS encryption, hashing passwords, and preventing SQL injection attacks. The system should also prevent unauthorized access to the admin account.
5. Reporting:
   The system should provide reports on course enrollment and student activity for the admin to review.
6. User Interface:

The user interface should be user-friendly, modern, and easy to navigate. The system should have a responsive design that works on desktop and mobile devices.

Non-Functional Requirements:

1. Usability:

The system should be easy to use and navigate for both users and admin,

The user interface should be clean and intuitive.

2. Performance:

The system should be able to handle a large number of users and courses without any performance issues,

The response time of the system should be fast.

3. Security:

The system should have a secure authentication mechanism to prevent unauthorized access.

4. Compatibility:

The system should be compatible with different web browsers and devices.

The system should be responsive and adapt to different screen sizes.

5. Maintainability:

The system should be easy to maintain and update.

The system should be well-documented and easy to understand.

Technical Requirements

1. **Frontend Technologies**:

a. The frontend of the application should be built using the React JavaScript library.

b. The user interface should be responsive, modern, and accessible.

c. The frontend should communicate with the backend through RESTful API endpoints.

d. The frontend should use state management libraries like Redux or MobX.

2**. Backend Technologie**s:

a. The backend of the application should be built using Spring Boot, a Java-based framework.

b. The backend should provide RESTful API endpoints for the frontend to interact with.

c. The backend should use a database for storing user and course data, such as MySQL or PostgreSQL.

d. The backend should use Spring Security for authentication and authorization of users.

3. **Deployment**:

a. The application should be deployed on a cloud platform, such as AWS or Google Cloud.

b. The application should be containerized using Docker for easy deployment and scaling.

4. **Performance**:

a. The application should be optimized for performance, including fast load times and responsive user interactions.

b. The application should use caching mechanisms, such as Redis or Memcached, to reduce database queries.

c. The application should be load-tested to ensure it can handle a large number of concurrent users.

5. Security:

a. The application should use HTTPS encryption for all communication between the frontend and backend.

b. The application should use secure authentication mechanisms, such as JWT tokens or OAuth2.

c. The application should be protected against common web vulnerabilities, such as SQL injection and cross-site scripting (XSS) attacks.

d. The application should use monitoring tools, such as Prometheus and Grafana, to detect and respond to security incidents.

6. Maintenance:

a. The application should be easy to maintain and update, with clear documentation and version control using tools like Git.

b. The application should use logging and error reporting mechanisms, such as ELK stack or Sentry, to help diagnose and fix issues.

c. The application should be kept up to date with security patches and updates to libraries and frameworks.