Software Requirement Specification

Title:”SkillSphere: EdTech Learning Hub and Coding Compiler”

1. Introduction

The introduction section provides an overview of the EdTech Articles and Coding Compiler Platform, including its purpose, target audience, intended use, product scope, and key definitions.

1.1 Purpose

The purpose of the EdTech Articles and Coding Compiler Platform is to create a comprehensive and interactive web-based application that caters to the needs of software enthusiasts, developers, and learners. The platform offers a space for users to share their expertise through blogs, discuss interview experiences, and practice coding problems using an integrated coding compiler. By offering these features, the platform aims to foster knowledge sharing, collaboration, and skill development within the software community.

1.2 Intended Audience

The intended audience for the EdTech Blog and Coding Compiler Platform includes:
Software developers and enthusiasts seeking a platform to share their insights and learn from others.
Students and learners looking to enhance their programming skills and gain practical experience.
Interview candidates who want to share their interview experiences and learn from others' insights.
Tech enthusiasts interested in staying up-to-date with the latest software-related technologies.

1.3 Intended Use

The EdTech platform is designed to serve as a hub for software-related content, offering the following functionalities:
Creation and publication of blogs on a wide range of software technologies.
Sharing interview experiences to provide valuable insights into interview processes and questions.
Interactive coding compiler supporting C/C++, Python, Java, and JavaScript for practice and learning.
User-friendly interface for accessing, exploring, and engaging with technology-related content.

1.4 Product Scope

The scope of the EdTech Blog and Coding Compiler Platform includes the following features:
User registration and authentication using local credentials, Google, or Facebook.
Blog creation, editing, categorization, and deletion functionalities.
Sharing and categorization of interview experiences.
Coding compiler for C/C++, Python, Java, and JavaScript with code execution capabilities.

User dashboards displaying published content, drafts, and coding history.
Integration of Spring Security and JWT for secure user authentication.
Clear and comprehensive API documentation using Swagger-UI.
Utilization of H2 Database for development/testing and MySQL for production data storage.
Responsive and user-friendly interfaces developed using React 16+.
Integration of npm, yarn, and webpack for frontend build, and Maven for server build.

1.5 Definitions and Acronyms

JWT: JSON Web Token, a secure method for transmitting information between parties as a JSON object.
API: Application Programming Interface, a set of protocols and tools for building software applications.
JPA: Java Persistence API, a Java specification for accessing, managing, and persisting data between Java objects and relational databases.
H2 Database: An in-memory and disk-based Java database.
MySQL: An open-source relational database management system.
React: A JavaScript library for building user interfaces.
Spring Boot: An extension of the Spring framework that simplifies the process of building production-ready applications.

2. System Requirements

2.1 Hardware Requirements

The hardware requirements describe the necessary physical components and resources for the successful operation of the EdTech Blog and Coding Compiler Platform. These include:

Web Server Infrastructure: The application requires a web server capable of hosting the frontend and backend components. This server should have sufficient processing power and memory to handle incoming user requests and serve responses efficiently.

Storage Capacity: The server must have adequate storage space to store user-generated content, including blogs, interview experiences, and code submissions. Additionally, storage space is needed for database files and other application-related assets.

2.2 Software Requirements

The software requirements outline the software components that need to be installed and configured for the platform to function correctly:

Operating System: The platform should be designed to work seamlessly on multiple operating systems, including Windows, macOS, and Linux distributions.

Web Server: A reliable web server such as Apache or Nginx is required to manage incoming HTTP requests and serve the frontend and backend components to users' browsers.

Database Management System: The platform requires MySQL for production data storage. Additionally, H2 Database is used for development and testing purposes.

Java Runtime Environment (JRE): The backend of the platform is developed using Java and requires a compatible JRE (Java 11 or higher) to run the Spring Boot application.

Node.js and npm: Node.js is needed for frontend development, and npm (Node Package Manager) is used to manage JavaScript packages required by the frontend.

Web Browsers: The platform should be designed to be compatible with major web browsers, including Chrome, Firefox, Safari, and Edge. It should provide a consistent and functional experience across these browsers.

2.3 Network Requirements

The network requirements pertain to the networking aspects needed for users to access and interact with the platform:

Internet Accessibility: The platform should be accessible over the internet using standard HTTP and secure HTTPS protocols. This accessibility ensures that users can interact with the platform from various locations and devices.

Bandwidth: The network should have sufficient bandwidth to accommodate multiple concurrent users interacting with the platform. Adequate bandwidth is essential to ensure smooth user experience, quick content delivery, and responsiveness.

3. Functional Requirements

Functional requirements detail the specific features and interactions that the EdTech Blog and Coding Compiler Platform must offer to users. These requirements outline the actions users can perform and the expected outcomes. Here's a more detailed elaboration of the functional requirements:

3.1 User Authentication and Authorization

User Registration:

Users can register by providing essential information such as name, email, and password. Users can also choose to register using their Google or Facebook accounts.

Authentication:

Users can log in using their registered credentials or external accounts.
The system verifies user credentials and issues a JWT token upon successful authentication.
Authorization:

Different user roles (e.g., regular user, admin) determine access levels to platform features.
Administrators have additional privileges, such as managing user accounts and content.

3.2 Article Management

Articles Creation:

Authenticated users can create new blogs by entering a title, content, and selecting relevant categories.
Articles can be categorized based on technology topics (e.g., Web Development, Machine Learning).
Editing and Deletion:
Authors have the ability to edit and update their own blogs to make improvements or corrections.
Authors can also choose to delete their blogs if needed.
Rich Text Editing:
A rich text editor provides users with options to format text, insert images, and apply styles to enhance blog content.

3.3 Interview Experience Sharing

Sharing Experiences:

Authenticated users can share their interview experiences by providing details about the interview process, questions asked, and their own insights.
Users can offer advice and tips to others based on their experiences.
Categorization:

Users can categorize their interview experiences, making it easier for other users to find relevant content.
Categories may include Interview Tips, Technical Interviews, Behavioral Interviews, etc.

3.4 Coding Compiler

Language Support:

The coding compiler should support multiple programming languages, including C/C++, Python, Java, and JavaScript.

Users can select the language they want to code in.
Code Writing:

Users can write, edit, and format code within the coding environment provided by the platform.
Syntax highlighting and auto-indentation improve the coding experience.
Compilation and Execution:

Users can submit their code for compilation and execution.
The platform should display any errors or output generated by the code execution.

3.5 User Dashboard

Personalized Overview:

Each user has a dashboard that displays a summary of their published blogs, shared interview experiences, and coding history.
Users can quickly access their most recent activity from the dashboard.
Content Management:

Users can manage their drafts, edit published blogs, and delete content as needed.
A list of saved coding snippets helps users revisit and reuse their code.

3.6 API Documentation

Swagger-UI Integration:
API endpoints are documented using Swagger-UI, providing users and developers with a clear understanding of available interactions.
Descriptions, parameters, request/response formats, and examples are provided for each API endpoint.

4. External Interface Requirements

External interface requirements define how the EdTech Blog and Coding Compiler Platform interacts with external systems, users, and devices. These requirements ensure seamless integration, user-friendly interactions, and effective communication. Here's an elaboration of the external interface requirements:

4.1 User Interfaces

The user interface (UI) of the platform is developed using React 16+.
The UI should provide a responsive design, adapting to various screen sizes and devices (desktop, tablet, mobile).

Users should have intuitive navigation and access to platform features through clearly labeled menus, buttons, and links.
The rich text editor for blog content should offer familiar text formatting options (e.g., bold, italic, bullet points) and image insertion.

4.2 Authentication Interfaces

Users should have a login and registration page where they can enter their credentials or choose to log in using external accounts (Google, Facebook).
The system should display appropriate error messages for incorrect login attempts or registration issues.
After successful authentication, users receive JWT tokens, which are sent with subsequent requests to authenticate and authorize them.

4.3 API Interfaces

The backend server, built with Spring Boot, exposes API endpoints that interact with the frontend and client applications.
API endpoints should follow RESTful principles, using clear and meaningful URLs to access different functionalities.
Swagger-UI integration provides API documentation accessible to both developers and users, including descriptions, request/response formats, and examples.

4.4 Database Interfaces

The application interfaces with the H2 Database during development/testing and MySQL for production data storage.
Database queries and operations should follow best practices to ensure efficient data retrieval and manipulation.
Secure database connection parameters (e.g., connection strings, credentials) should be stored securely and managed appropriately.

4.5 Coding Compiler Interface

The coding compiler interface provides a coding environment where users can write, compile, and run code.
Users should have input fields for code entry and a submit button to initiate compilation and execution.
Output from the compiler and runtime should be displayed clearly, indicating success or errors in the code execution.

4.6 Social Media Interfaces

Users can choose to authenticate using their Google or Facebook accounts, requiring integration with their respective authentication APIs.

Users should have the option to share their published blogs and interview experiences on social media platforms through share buttons or links.

4.7 Web Server Interfaces

The frontend and backend components of the platform are hosted on a web server (e.g., Apache, Nginx).
The web server should be configured to route incoming HTTP/HTTPS requests to the appropriate backend API endpoints or static frontend files.

4.8 Third-Party Services

External services like Google and Facebook authentication APIs should be integrated to facilitate third-party account authentication.
If the platform uses payment gateways for premium features, integration with those gateways would be required.

4.9 Web Browser Interfaces

The platform should be accessible and fully functional on modern web browsers such as Chrome, Firefox, Safari, and Edge.
Compatibility testing ensures consistent behavior and appearance across different browsers.

5. Non-Functional Requirements (NFRs)

Non-functional requirements define the qualities and characteristics that the EdTech Blog and Coding Compiler Platform must possess to ensure its performance, security, usability, and overall user experience. These requirements focus on aspects beyond specific features and interactions. Here's a more detailed elaboration of the non-functional requirements:

5.1 Performance

Responsiveness: The platform should load quickly and respond promptly to user interactions, ensuring a smooth and seamless experience.
Scalability: The application architecture should support horizontal scaling to accommodate increased user traffic without degradation in performance.
Code Compiler Performance: The coding compiler should execute code efficiently and provide quick feedback to users.

5.2 Security

Data Protection: User data, including personal information and authentication details, should be securely stored and encrypted to prevent unauthorized access.

Secure Authentication: JWT-based authentication should be implemented securely to prevent token-based attacks and unauthorized access.
Secure Coding Practices: Development should follow secure coding practices to prevent vulnerabilities such as SQL injection, XSS attacks, and CSRF attacks.
Role-Based Access Control: Authorization mechanisms should ensure that users can only access functionalities and data based on their roles.

5.3 Usability

Intuitive UI/UX: The user interface should be designed intuitively, with clear navigation and user-friendly interactions.
Consistency: Design elements, terminology, and layout should be consistent throughout the platform for a unified user experience.
Accessibility: The platform should adhere to accessibility standards, ensuring that it's usable by individuals with disabilities.

5.4 Reliability

Uptime and Availability: The platform should aim for high availability, minimizing downtime and disruptions.
Error Handling: The system should handle errors gracefully, providing users with meaningful error messages and resolutions.
Backup and Recovery: Regular data backups and a robust disaster recovery plan should be in place to prevent data loss.

5.5 Compatibility

Cross-Browser Compatibility: The platform should be fully functional and visually consistent across major web browsers.
Device Compatibility: The UI should adapt and work well on different devices, including desktops, tablets, and mobile phones.

5.6 Maintainability

Modular Architecture: The application should follow a modular architecture, making it easier to maintain and update individual components.
Documentation: Code, APIs, and system architecture should be well-documented for easier troubleshooting, maintenance, and onboarding of new developers.
Code Quality: Development should adhere to coding standards, ensuring readability, maintainability, and reducing technical debt.