Assignment No : B5

Software Requirements Specification (SRS) for Code Sync

Said Prachi Sachin

1 Introduction

1.1 Purpose

This document outlines the software requirements for **Code Sync**, a real-time collabora- tive code editing platform that enables multiple users to edit code simultaneously, engage in group chats, and collaborate on sketches and drawings. The goal is to create a seam- less experience for remote development teams, educational purposes, and collaborative coding sessions.

1.2 Scope

Code Sync will provide a web-based interface allowing users to:

- Edit code collaboratively in real-time.
- Communicate via real-time group chat.

1.3 Definitions, Acronyms, and Abbreviations

- **Real-Time Collaboration**: The ability for multiple users to interact and make changes to the code or drawing canvas simultaneously.
- **Socket.io**: A JavaScript library for real-time web applications, enabling bidirectional communication between web clients and servers.
- **Frontend**: The client-side part of the application that interacts with the user, built with React, React Router, and Tailwind CSS.
- **Backend**: The server-side part that handles data processing, storage, and real-time synchronization, built with Node.js.

1.4 References

- React Documentation: https://reactjs.org/
- Node.js Documentation: https://nodejs.org/
- Socket.io Documentation: https://socket.io/
- Tailwind CSS Documentation: https://tailwindcss.com/

2 Overall Description

2.1 Product Perspective

Code Sync is a new, standalone web application designed to facilitate real-time code editing and collaborative drawing for developers and educators. It will use a client-server architecture where the client (browser) communicates with the server using WebSockets (Socket.io).

2.2 Product Functions

The main functionalities of **Code Sync** include:

- **Collaborative Code Editor**: Supports multiple programming languages, pro- vides syntax highlighting, line numbering, and code folding. Allows multiple users to edit code files simultaneously and supports version control features like undo/redo and file history.
- **Real-Time Group Chat**: Enables text-based communication between all con- nected users, displays messages in a chat window within the application, and notifies users of new messages.
- **Collaborative Drawing/Sketching**: Offers a drawing canvas where multiple users can draw and sketch in real time, provides basic drawing tools (pen, eraser, shapes, colors), and supports saving and exporting drawings.

2.3 User Characteristics

The target users include:

- Software development teams.
- Educational instructors and students.
- Open-source contributors.
- Anyone needing a real-time collaborative coding environment.

2.4 Constraints

- **Performance**: The platform must handle up to 100 concurrent users per session with minimal latency (¡100ms).
- **Security**: Must implement secure user authentication, data encryption, and pro- tection against common web vulnerabilities.
- **Technology Stack**: Limited to React, React Router, Tailwind CSS for the fron- tend, and Node.js with Socket.io for the backend.

2.5 Assumptions and Dependencies

- Users will have a stable internet connection.
- The platform will run on modern web browsers (Chrome, Firefox, Edge, Safari).
- The backend server will have adequate resources to handle concurrent users.

3 Specific Requirements

3.1 Functional Requirements

3.1.1 Collaborative Code Editor

- **FR1.1**: The system shall support multiple programming languages (e.g., JavaScript, Python, Java).
- **FR1.2**: The system shall allow multiple users to edit code simultaneously with real-time synchronization.
- **FR1.3**: The system shall provide syntax highlighting, line numbering, and code folding.
- **FR1.4**: The system shall allow users to undo/redo changes.
- **FR1.5**: The system shall save code files automatically at regular intervals and upon user request.
- **FR1.6**: The system shall maintain a version history of code changes.

3.1.2 Real-Time Group Chat

- **FR2.1**: The system shall provide a text-based chat feature for users to communicate in real time.
- **FR2.2**: The system shall display a list of currently active users.
- **FR2.3**: The system shall provide message notifications.

3.1.3 Collaborative Drawing/Sketching

- **FR3.1**: The system shall provide a drawing canvas that allows multiple users to draw and sketch in real time.
- **FR3.2**: The system shall offer basic drawing tools, including a pen, eraser, shape tools, and color picker.
- **FR3.3**: The system shall allow users to save and export their drawings.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

- **NFR1.1**: The system shall synchronize changes in less than 100ms for up to 100 concurrent users.
- **NFR1.2**: The system shall load the main application interface within 3 seconds.

3.2.2 Security Requirements

- **NFR2.1**: The system shall use HTTPS for all data transmissions.
- **NFR2.2**: The system shall implement secure user authentication (e.g., OAuth2.0).
- **NFR2.3**: The system shall encrypt all user data, including messages and code files, at rest and in transit.

3.2.3 Usability Requirements

- **NFR3.1**: The system shall have an intuitive user interface designed using Tailwind CSS.
- **NFR3.2**: The system shall provide tooltips and help documentation for all major features.

3.2.4 Scalability Requirements

- **NFR4.1**: The system shall support up to 500 concurrent users across multiple sessions.
- **NFR4.2**: The system shall be designed to allow scaling by adding more server instances.

3.3 Interface Requirements

3.3.1 User Interface

- The application shall have a responsive design compatible with desktops, tablets, and mobile devices.
- The main interface shall include sections for the code editor, real-time chat, and drawing/sketching tools.

3.3.2 API Interface

- The backend shall expose RESTful APIs for managing user sessions, saving code and drawings, and handling user authentication.

4 External Interface Requirements

4.1 Hardware Interfaces

- The client-side application must run on devices with at least 2GB of RAM and a dual-core processor.
- The server must be hosted on a machine with at least 8GB of RAM and a quad-core processor.

4.2 Software Interfaces

- The frontend shall interact with the bac1-28,31-49kend using REST APIs and WebSocket connections.

4.3 Communication Interfaces

- The application shall communicate over TCP/IP using secure WebSockets (wss://) for real-time collaboration and HTTPS for API calls.

5 Other Requirements

- **Deployment**: The application shall be deployed on a cloud platform like AWS or Azure.
- **Logging and Monitoring**: The system shall log all user activities and provide monitoring tools to detect and respond to issues.

6 Appendices

- **Appendix A**: User Interface Mockups (To be developed).
- **Appendix B**: Database Schema (To be developed).