A

Project Report

On

Real-Time Online IDE

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Rohit Tiwari	**2261494**
Anish Kumar	**2261049**
Arnav Singh	**2261049**
Himanshu Singh	**2261273**

Under the Guidance of

Mr.Anubhav Bewerwal

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI DISTRICT- NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, **Rohit Tiwari, Anish Kumar, Arnav Singh, Himanshu Singh** hereby declare the work, which is being presented in the project, entitled **'Real-Time Online IDE'** in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of my work carried out under the supervision of **Mr. Anubhav Bewerwal, Assistant Professor.** The matter embodied in this project has not been submitted by me for the award of any other degree.

Date:

Rohit Tiwari
Anish Kumar
Arnav Singh
Himanshu Singh

Graphic Era Hill University
BHIMTAL CAMPUS

CERTIFICATE

The term work of Project Based Learning, being submitted by Rohit Tiwari (2261494), Anish Kumar (2261049), Arnav Singh (2261049), and Himanshu Singh (2261273) to Graphic Era Hill University Bhimtal Campus for the award of Bonafide work carried out by us. They have worked under my guidance and supervision and fulfilled the requirements for the submission of this work report.

(Mr.Anubhav Bewerwal)

Faculty-in-Charge

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director, **Prof. (Col.) Anil Nair (Retd.)**, GEHU Bhimtal Campus, for permitting us to carry out this project work under his excellent and optimistic supervision. This has all been possible due to his inspiring leadership, able guidance, and useful suggestions that helped us grow as creative researchers and complete the research work on time.

Words are inadequate to express our gratitude to **God** for providing us with everything we needed throughout this journey. We would also like to extend our heartfelt thanks to our President, **Prof. (Dr.) Kamal Ghanshala**, for providing us with the necessary infrastructure and facilities—without which this work would not have been possible.

We express our sincere gratitude to **Dr. Ankur Singh Bisht** (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide **Mr. Anubhav Bewerwal** (Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus), and other faculty members for their insightful comments, constructive suggestions, and the time they devoted to reviewing this report.

Finally, yet importantly, we would like to express our heartfelt thanks to our beloved parents for their unwavering moral support, affection, and blessings. We also extend our sincere thanks to all our friends and well-wishers for their help and encouragement in the successful completion of this project.

Rohit Tiwari, 2261494

Anish Kumar, 2261049

Arnav Singh, 2261049

Himanshu Singh, 2261273

Abstract

In the rapidly evolving landscape of remote collaboration and online development, the need for real-time, web-based coding environments has become increasingly significant. This project, titled **Code-Sync**, presents the design and implementation of a web application that enables multiple users to write and edit code simultaneously in real-time. The platform leverages modern web technologies including **React.js** for the frontend, **Node.js** for the backend, and **Socket.IO** for real-time bi-directional communication.

Code-Sync is built with the primary objective of facilitating seamless code sharing and collaboration, particularly useful in scenarios such as online coding interviews, peer programming, technical education, and hackathons. The application features a collaborative code editor powered by **CodeMirror**, which supports syntax highlighting and provides a user-friendly interface for editing code live.

The project explores various aspects of full-stack development, including client-server communication, component-based UI design, and WebSocket-based messaging. It also addresses the challenges of maintaining synchronization, handling multiple users, and creating a responsive and intuitive user experience.

While the current implementation focuses on core functionality, several limitations exist, such as the absence of authentication, persistent storage, and code execution features. These limitations are acknowledged, and future enhancements have been proposed to transform Code-Sync into a robust and scalable collaborative development tool.

This project not only fulfills academic requirements but also demonstrates a practical application of real-time web technologies, offering valuable insights into modern software development practices.

TABLEOFCONTENTS

CHAPTER 1: INTRODUCTION

1.1 Prologue

In the era of cloud computing, real-time collaboration has become a foundational necessity across various domains—especially in software development. Traditional code editors lack the ability to provide synchronized, live collaboration features that modern teams require. Addressing this gap, Code-Sync is a web-based real-time collaborative code editor that allows multiple users to write, edit, and sync code simultaneously. This project merges the functionality of a conventional code editor with the real-time interactivity of collaborative platforms using modern web technologies such as React.js, Node.js, and Socket.IO.

1.2 Background and Motivations

Remote collaboration has experienced a significant surge, especially in the post-pandemic era, as developers increasingly require tools that enable seamless, real-time interaction. While platforms like Google Docs have set a high standard for document collaboration, similar capabilities in code editing remain underdeveloped. Most existing collaborative coding tools are either proprietary, resource-intensive, or not beginner-friendly, leaving a gap in accessible solutions. This project was driven by the need to address that gap by creating a lightweight, open-source, and easy-to-deploy collaborative coding platform. Additionally, it serves as an opportunity to learn and apply core concepts of full-stack development using modern technologies, while also gaining a deeper understanding of real-time event handling through the use of WebSockets.

1.3 Problem Statement

Despite the availability of numerous code editors and IDEs, there remains a noticeable lack of simple, efficient, and easily accessible tools that enable multiple developers to collaborate on the same codebase in real-time. In many cases, developers are forced to rely on screen sharing or third-party applications that are not specifically tailored for collaborative coding, leading to inefficiencies and a disjointed development experience. **Code-Sync** is designed to address these challenges by providing a dedicated platform for real-time code collaboration. It aims to solve the absence of native real-time syncing features in popular editors, streamline collaboration during online pair programming or group projects, and simplify the setup and management of real-time code-sharing environments.

1.4 Objectives and Research Methodology

The primary objective of this project is to develop a real-time collaborative code editor using modern web technologies that ensures smooth and efficient multi-user interaction. A key focus is on implementing robust multi-user connectivity with seamless syncing of code changes to enable real-time collaboration. The platform is designed with a minimal and intuitive user interface to ensure ease of use and smooth interaction, even for beginners. Additionally, the system follows a modular structure, allowing for future enhancements such as code execution, customizable themes, and additional language support to be integrated effortlessly, ensuring long-term scalability and adaptability.

The research methodology for this project began with a comprehensive literature review, analyzing existing platforms such as CodePen, Replit, and Google Docs to understand how real-time collaboration is implemented and managed. This was followed by a thorough evaluation of suitable technologies, leading to the selection of React.js for the frontend, Node.js for the backend, and Socket.IO for enabling WebSocket-based real-time communication. With the technology stack in place, a basic prototype supporting two users was initially developed to validate the core functionality. This prototype was gradually scaled to accommodate multiple clients, ensuring broader usability. The final phase involved rigorous testing with dummy users to simulate real-world collaboration scenarios, during which the synchronization logic was refined and optimized based on observed performance and user experience.

1.5 Project Organization

The Code-Sync project is organized into the following key parts:

Module	Description
server.js	Starts the backend server and handles WebSocket connections.
src/socket.js	Manages real-time messaging between users using Socket.IO.
src/Actions.js	Defines custom event types for WebSocket communication.
src/App.jsx	Main entry component for the React frontend application.
src/index.js	React app entry point, renders the main component.
src/components/Editor.jsx	Code editor component using CodeMirror.
src/components/Client.jsx	Manages and displays connected client/user list.
src/components/CodeRun.jsx	Displays run/execution section (or preview area).
public/index.html	Root HTML template rendered by the React app.
public/manifest.json	Configures PWA (Progressive Web App) settings.
public/logo192.png, logo512.png	Logos used in the frontend and manifest.
src/pages/	(If present) Contains route-based components or pages.
src/App.css, index.css	Contains frontend styling and theme configurations.
.gitignore	Specifies files/folders to ignore in version control.
package.json, package-lock.json	Defines project dependencies and scripts for build, start, etc.
README.md	Project description, usage instructions, and setup guide.

CHAPTER 2: PHASES OF SOFTWARE DEVELOPMENT CYCLE

Software development typically follows a series of systematic steps to ensure the successful completion and deployment of a project. The development of the Code-Sync project was also carried out following a simplified Software Development Life Cycle (SDLC), comprising of the following major phases:

Phases Followed:

1. **Requirement Analysis**
   Understanding the need for a collaborative code editor and identifying the required features such as real-time syncing, multi-user sessions, and a simple UI.

2. **System Design**
   Designing the architecture involving frontend (React), backend (Node.js), and real-time communication (Socket.IO). Component hierarchy and routing were planned in this phase.

3. **Implementation**
   Developing the frontend components, backend server, WebSocket logic, and integrating everything into a functional full-stack app.

4. **Testing**
   Functionality was tested using multiple browser sessions to simulate real-time collaboration and user behavior.

5. **Deployment & Maintenance**
   The app was prepared for deployment using Node.js and can be hosted on cloud platforms like Vercel, Render, or Heroku.

2.1 Hardware Requirements

Component	Minimum Specification
Processor	Intel Core i3 (or equivalent)
RAM	4 GB
Hard Disk	250 MB of free space
Monitor	720p or higher resolution
Input Devices	Keyboard and Mouse
Internet Connection	Required for real-time synchronization

2.2 Software Requirements

Software	Details
Operating System	Windows 10 / Linux / macOS
Node.js	v14 or higher (for backend and dependency management)
NPM	Comes with Node.js (used for installing packages)
Web Browser	Chrome / Firefox / Edge (latest version recommended)
Code Editor	Visual Studio Code or any modern IDE
Git	For version control
React.js	Frontend JavaScript framework (via `create-react-app`)
Socket.IO	WebSocket library for real-time communication
Any Terminal/Command Line	For running the local server

CHAPTER 3: CODING OF FUNCTIONS

The Code-Sync project integrates various technologies such as React.js, Node.js, and Socket.IO to enable real-time collaborative code editing. This chapter describes the core logic and functionality of the system and how different modules work together to achieve the project's objectives.

3.1 Backend Functionality (server.js)

The backend is powered by Node.js and Socket.IO, enabling real-time communication between users.

Key Functions:

- **Starting the server:**

```
const express = require('express');
const http = require('http');
const { Server } = require('socket.io');
const app = express();
const server = http.createServer(app);
const io = new Server(server);
```

- **Handling socket connections:**

```
io.on('connection', (socket) => {
    socket.on('join', ({ roomId, username }) => {
        socket.join(roomId);
        // Broadcast new user to others
    });

    socket.on('code-change', ({ roomId, code }) => {
        socket.to(roomId).emit('code-update', code);
    });

    socket.on('disconnect', () => {
        // Handle disconnection
    });
});
```

3.2 Frontend Functionality (src/)

The frontend is built using **React.js**, which provides a responsive interface and dynamic component rendering.

Editor.jsx – Code Editor Component:

- Integrates CodeMirror for syntax highlighting.
- Handles local code updates and emits changes to the server:

```
const handleChange = (editor, data, value) => {
    setCode(value);
    socketRef.current.emit('code-change', { roomId, code: value });
};
```

Client.jsx – Connected Users:

- Displays all active users in the current room.
- Updates when users join or leave.

CodeRun.jsx – Code Output (optional future enhancement):

- Placeholder for code execution or output display.

App.jsx – Main Application Router:

- Manages routing between the homepage and editor page.
- Passes required props (e.g., roomId, username) between components.

3.3 WebSocket Logic (socket.js)

Handles real-time communication**:** The socket.js file is responsible for managing real-time communicationbetween the client and the server. It begins by establishing a WebSocket connection from the client side using a library like Socket.IO.

- Establishes the client-side socket connection:

```
const socket = io('http://localhost:3001');
```

- Listens and responds to events:

```
socket.on('code-update', (code) => {
    // Update code in the editor
});
```

3.4 Utilities and Event Constants (Actions.js)

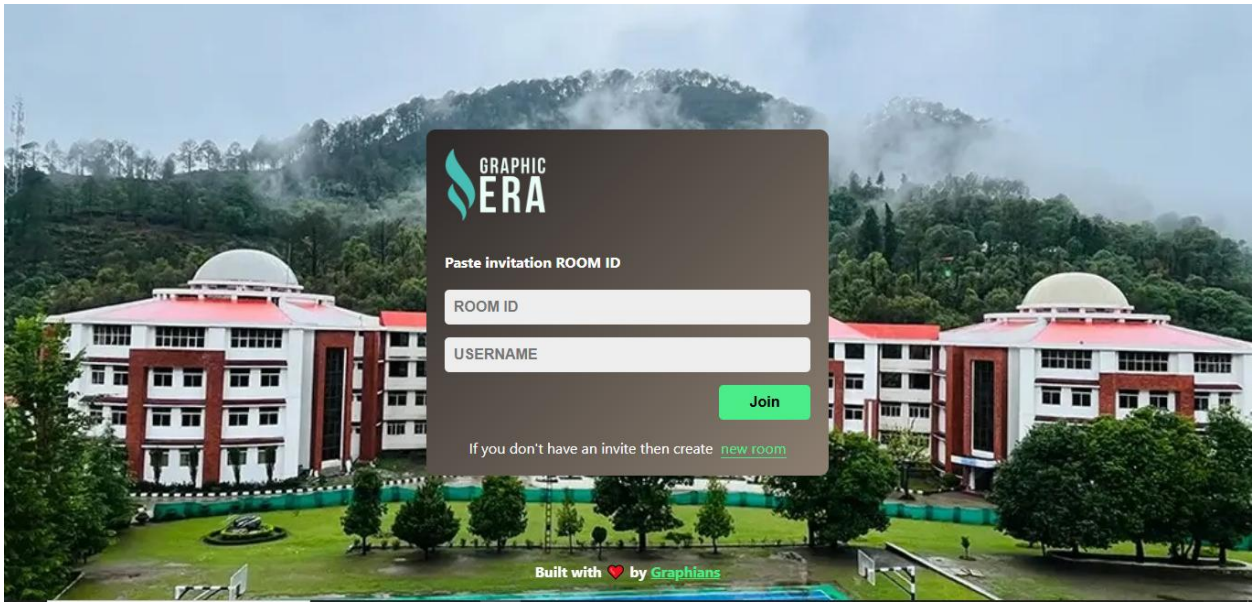
Defines string constants used to emit and listen to socket events, making the codebase cleaner and less error-prone:

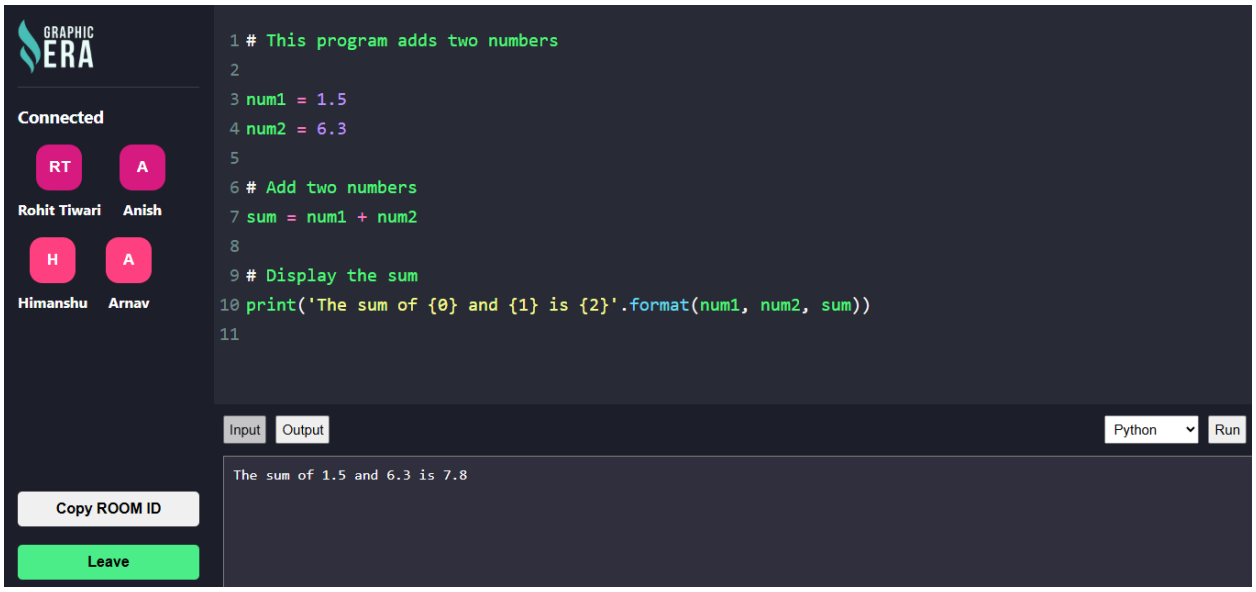
```
export const CODE_CHANGE = 'code-change';
export const JOIN = 'join';
export const LEAVE = 'leave';
```

3.5 Supporting Files

- **public/index.html:** The main HTML page into which React renders components.
- **index.js:** Entry point for the React app, renders `<App />`.
- **App.cssandindex.css:**Contain styles for theme, layout, and UI responsiveness.

CHAPTER 4: SNAPSHOTS





CHAPTER 5: LIMITATIONS

While the Code-Sync project effectively achieves its goal of enabling real-time collaborative coding, it is important to recognize several limitations that currently exist in its implementation. These constraints highlight areas for improvement and future development.

5.1 No Code Execution Support

Although the editor allows collaborative writing of code, it does not include functionality to compile or run code within the platform. This limits its utility to purely collaborative writing rather than testing or debugging.

5.2 No Syntax Error Highlighting

The integrated CodeMirror editor provides basic syntax highlighting, but there is no real-timeerror detection or linting to help identify mistakes in the code.

5.3 No Authentication or User Roles

Currently, the system lacks a secure login or user authentication mechanism. Anyone with a room ID can join and edit code, which may lead to misuse or unauthorized access.

5.4 Limited Scalability

The application is designed for small-scale collaboration. With more users in a single session or multiple rooms, server performance may degrade due to increased socket traffic.

5.5 No Persistent Storage

There is no option to save or load previous sessions. Once the session ends or the page is refreshed, all code is lost unless manually copied and saved.

5.6 No Voice/Video/Chat Integration

Effective real-time collaboration often requires communication tools. This version of Code-Sync does not support voice, video, or even a basic text chat feature.

5.7 UI/UX Limitations

Although functional, the current user interface is minimalistic and may not be user-friendly for non-technical users. Features like theming, drag-drop, and responsive design need enhancement.

CHAPTER 6: ENHANCEMENTS

The current implementation of the Code-Sync project serves as a functional prototype for real-time collaborative code editing. However, various enhancements can be introduced to improve its scalability, usability, and feature set. Below are several proposed enhancements that can be implemented in future iterations:

6.1 Code Execution Feature

Integrate a code execution engine (such as Judge0 or a custom backend using Docker containers) that allows users to compile and run code directly within the editor. This would make the platform more practical for learning, testing, and debugging.

6.2 User Authentication System

Add a secure login and registration system using technologies such as JWT (JSON Web Tokens), Firebase Auth, or OAuth. This will restrict access, prevent unauthorized edits, and allow personalized experiences.

6.3 Persistent Storage

To enhance the functionality and user experience of the application, implementing persistent storage using a database such as MongoDB, Firebase, or PostgreSQL is essential. This integration allows the system to save user sessions and project data securely, ensuring that progress is not lost.

6.4 Role-Based Permissions

Introduce role management, allowing certain users to have “editor” roles while others are “viewers.” This can help manage sessions during collaborative teaching, interviews, or group discussions.

6.5 Chat/Communication Panel

Add a real-time chat feature within each room to facilitate collaboration. Optionally, integrate video/audio conferencing APIs (like WebRTC or Zoom SDK) for live discussions.

6.6 Syntax Checking and Linting

Integrate linters or static analyzers to detect syntax and logical errors in real-time.

6.7 Theme Customization

Allow users to switch between multiple editor themes (light, dark, high-contrast) and personalize the UI for better accessibility and comfort.

6.8 Conflict Resolution Logic

To maintain code integrity during collaboration, the editor should include conflict resolution mechanisms such as line locking. This prevents multiple users from editing the same line at the same time, reducing the risk of overwriting each other's changes. For more advanced setups, real-time merging techniques like operational transformation or CRDTs can be used to handle simultaneous edits more efficiently.

6.9 Mobile & Tablet Support

Making the UI responsive and touch-friendly ensures that users can collaborate from tablets and smartphones. This involves adapting layouts, resizing elements, and optimizing touch controls so that the platform remains easy to use on smaller screens, allowing users to work from anywhere.

6.10 Deployment on Scalable Infrastructure

Deploying the app on cloud platforms like Heroku, Vercel, or AWS allows it to scale based on traffic. Features like load balancing and real-time database syncing help maintain performance and stability, especially for large teams or global users collaborating in real time.

CHAPTER 7: CONCLUSION

The Code-Sync project was developed with the objective of creating a real-time collaborative code editing platform that allows multiple users to write and edit code simultaneously. Built using React.js for the frontend, Node.js for the backend, and Socket.IO for real-time communication, the application successfully demonstrates the core functionality of collaborative development tools used in modern software engineering environments.

Through this project, we were able to explore the integration of WebSockets, client-server architecture, modular frontend components, and the dynamic rendering of user interfaces. The system provides a functional base that can be expanded with advanced features like code execution, persistent storage, and user authentication.

While there are limitations in the current version—such as the lack of code execution, no authentication, and no permanent data storage—this project serves as a strong foundation for a more complex, scalable solution. The suggested enhancements can transform Code-Sync into a fully-featured online collaborative coding platform suitable for educational, interview, and team development purposes.

In conclusion, this project not only fulfills the academic requirement but also provides practical exposure to full-stack development and real-world application design. It reflects a meaningful step toward understanding and building modern web-based development tools.

The development of the **Code-Sync** project has been an insightful and practical journey into the world of real-time web applications, full-stack development, and collaborative software tools. The goal of this project was to design and implement a web-based platform where multiple users can write, edit, and share code in real-time. With the increasing need for remote collaboration, such tools have gained significant importance in both educational and professional contexts. The successful implementation of this project marks an important milestone in understanding modern software engineering practices.

In conclusion, **Code-Sync** is a powerful demonstration of how modern web technologies can be combined to build collaborative, real-time applications. It reflects the shift from static applications to dynamic, user-centric solutions that promote interactivity and productivity. Beyond being a semester project, Code-Sync serves as a base upon which more advanced and robust collaborative tools can be developed.

This project has not only enhanced our technical skills but also improved our understanding of teamwork, project planning, and real-world software development practices. It stands as a testament to the possibilities of innovation when theory meets practice.

REFERENCES

1. Zhang, Y., & Zhang, X. (2015). "A Study on the Performance of Multithreaded Web Servers." Journal of Computer and System Sciences, 81(2), 371–387. DOI: 10.1016/j.jcss.2014.11.012

2. Finkel, H. (2009). "Designing High-Performance Web Servers: A Study on the Impacts of Concurrency." IEEE Transactions on Parallel and Distributed Systems, 20(6), 889–896. DOI: 10.1109/TPDS.2009.36

3. Kumar, R., & Ghosh, A. (2017). "Concurrency Control Techniques in Multithreaded Web Servers." International Journal of Computer Applications, 164(1), 37–42. DOI: 10.5120/ijca2017915637

4. Parker, B. (2020). Concurrency and Multithreading in Web Server Design. O'Reilly Media.

5. Gao, W., & Huang, J. (2013). "A Survey on Web Server Technologies." Journal of Computer Science and Technology, 28(1), 31–49. DOI: 10.1007/s11390-013-1308-0

6. Davis, M., & White, T. (2016). "Using Thread Pools in Multithreaded Web Applications." ACM Transactions on the Web, 10(4), Article 15. DOI: 10.1145/3028651

7. Tanenbaum, A. S., & Austin, T. (2012). Operating Systems: Design and Implementation (3rd ed.). Prentice Hall.

8. Duffy, J. (2018). Programming the Microsoft Bot Framework: A Multiplatform Approach to Building Chatbots. Microsoft Press.

9. Membrey, P., Plugge, E., & Hawkins, T. (2010). MongoDB: The Definitive Guide. O'Reilly Media.

10. Banks, A., & Porcello, E. (2020). *Learning React: Functional Web Development with React and Redux*. O'Reilly Media.

11. Raj, P., & Raman, A. (2019). "A Study on Real-Time Web Technologies using WebSockets." International Journal of Web & Semantic Technology (IJWesT), 10(1), 15–23. DOI: 10.5121/ijwest.2019.10102

12. CodeMirror. (2022). CodeMirror Documentation. Retrieved from https://codemirror.net/doc/manual.html

13. Node.js Foundation. (2021). Node.js Documentation. Retrieved from https://nodejs.org/en/docs

14. Sockets.IO. (2021). Socket.IO Official Documentation. Retrieved from https://socket.io/docs/