B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institute, Affiliated to VTU)

Bull Temple Road, Basavanagudi, Bengaluru - 560019

A Project Report on

"Collab Dev"

Submitted in partial fulfilment of the requirements for the award of degree

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

By

Aniket V Korwar 1BM23IS403

Rohan Raju Navalyal 1BM22IS162

Suprit R Sanadi 1BM23IS418

Under the guidance of

Dr. B S Mahalakshmi

Associate Professor

Department of Information Science and Engineering

2024-2025

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institute, Affiliated to VTU)

Bull Temple Road, Basavanagudi,

Bengaluru – 560019

Department of Information Science and Engineering

C E R T I F I C A T E

This is to certify that the project entitled "**Collab Dev**" is a bona-fide work carried out by **Aniket V Korwar (1BM23IS403), Rohan Raju Navalyal (1BM22IS162), Suprit R Sanadi (1BM23IS418)** in partial fulfilment for the award of degree of Bachelor of Engineering in **Information Science and Engineering** from **Visvesvaraya Technological University, Belgaum** during the year **2024-2025**. It is certified that all corrections/suggestions indicated for Internal Assessments have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering Degree.

Dr. B S Mahalakshmi	**Dr. Nalini M K**	**Dr. Bheemsha Arya**
Associate Professor	**Associate Professor and HOD**	**Principal**

Examiners

Name of the Examiner	**Signature of the Examiner**
1.	
2.	

DECLARATION

Aniket V Korwar (1BM23IS403), Rohan Raju Navalyal (1BM22IS162), Suprit R Sanadi (1BM23IS418) students of B.E. Information Science and Engineering, B.M.S. College of Engineering, Bangalore - 19, hereby declare that the capstone project entitled **"Collab Dev"** is an authentic work carried out under the supervision and guidance of **Dr. B S Mahalakshmi**, Associate Professor, Department of Information Science and Engineering, B.M.S. College of Engineering, Bangalore. We have not submitted the matter embodied to any other university or institution for the award of any other degree.

Place:

Date:

Name	USN	Signature
Aniket V Korwar	1BM23IS403	……...
Rohan Raju Navalyal	1BM22IS162	……...
Suprit R Sanadi	1BM23IS418	……...

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this Project Phase-1 would be incomplete without the mention of the people who made it possible through constant guidance and encouragement.

We would take this opportunity to express our heart-felt gratitude to **Dr. B. S. Ragini Narayan**, Chairperson, Donor Trustee, Member Secretary & Chairperson, BMSET. **Dr. P. Dayananda Pai,** Member Life Trustee, BMSET and **Dr. Bheemsha Arya**, Principal, B.M.S. College of Engineering for providing the necessary infrastructure to complete this Capstone Project Phase-1.

We wish to express our deepest gratitude and thanks to **Dr. Nalini M K,** Head of the Department, Information Science and Engineering.

We wish to express sincere thanks to our guide **Prof. Dr. B S Mahalakshmi,** Associate Professor, Department of Information Science and Engineering for helping us throughout and guiding us from time to time.

A warm thanks to all the faculty of Department of Information Science and Engineering, who have helped us with their views and encouraging ideas.

Aniket V Korwar (1BM23IS403)

Rohan Raju Navalyal (1BM22IS162)

Suprit R Sanadi (1BM23IS418)

ABSTRACT

Collab Dev is a web-based collaborative coding platform designed to enhance real-time programming experiences for developers, students, interviewers, and remote teams. The platform enables multiple users to simultaneously write, edit, and view code within a shared environment, fostering seamless cooperation. Collab Dev integrates modern tools such as the Monaco Editor, Socket.io for real-time communication, and Clerk Authentication for secure user access.

Key features include live code synchronization, a dynamic file explorer, and terminal emulators, which collectively offer an experience comparable to native development environments. Users can join sessions either through authenticated login or anonymously via a room code, with interaction privileges restricted to authorized collaborators to ensure privacy and control.

The application adopts a modular architecture with support for custom themes (light/dark modes) and aligns with modern UI practices using ShadCN UI and Tailwind CSS. The aim of Collab Dev is to bridge the gap between code sharing and real-time collaboration, making it particularly well-suited for coding bootcamps, technical interviews, pair programming, and remote development sessions.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

The field of software development is undergoing rapid transformation, especially with the increase in distributed teams, virtual learning environments, and remote work culture. Traditional development tools, though powerful, are often isolated and not built for real-time, multi-user collaboration. This creates friction for developers working in pair programming environments, educators teaching live coding sessions, and interviewers assessing candidates' coding skills.

To address this gap, Collab Dev introduces a browser-based collaborative coding platform that facilitates simultaneous code editing, real-time communication, terminal interaction, and dynamic project file navigation—all within a unified interface. By leveraging the latest web technologies, Collab Dev simplifies the development experience, making it accessible, efficient, and highly collaborative for modern-day use cases.

The platform eliminates the need for complex installations or plugin dependencies and provides a smooth, zero-configuration environment for its users. This enables a wide array of use cases ranging from virtual classrooms and coding bootcamps to technical assessments and collaborative team development.

1.2 Scope

The scope of the Collab Dev project includes:

- Development of a collaborative IDE that runs entirely in the browser.
- Real-time code synchronization between multiple users using WebSockets.
- A robust role-based access model allowing users to join as editors or viewers.
- Integration of a web-based terminal emulator for executing basic commands.
- A visual file explorer for navigating and managing project files.
- Authentication and session management via secure login or anonymous access using unique room codes.
- Support for multiple programming languages and themes (light/dark mode).
- Scalability for handling multiple collaborative sessions concurrently.
- UI built on modern libraries (Tailwind CSS, ShadCN UI) to ensure responsive design and accessibility.

1.3 Existing System

Currently available platforms for online coding and collaboration include:

- Visual Studio Code Live Share: A powerful collaborative tool but requires desktop installations, plugin configurations, and login accounts. Not suitable for spontaneous or guest-based collaboration.
- Replit: A web-based coding environment offering limited collaboration features. Free tiers have limitations on performance and real-time editing granularity.
- CodePen / JSFiddle: Frontend-focused tools without terminal or multi-file project support.
- Google Colab: Designed for Python notebooks and lacks full IDE functionalities.
- Screen-sharing tools (Zoom, Google Meet): These provide passive collaboration (observation only) and are not optimized for actual code manipulation.

The main limitations in existing systems are:

- Lack of native, in-browser collaborative development experiences.
- Limited terminal access and project file navigation.
- Inflexible session management and lack of guest-based viewing.
- Limited customization of UI and themes.

1.4 Proposed System

Collab Dev addresses the shortcomings of existing systems by offering:

- A real-time, browser-based collaborative IDE that eliminates setup time and installation dependencies.
- Multi-user editing with visual indicators for collaborative typing.
- Viewer-only mode for guests to securely observe a session.
- Secure sign-in options via Clerk, with session ownership and control features.
- An in-browser terminal using Xterm.js, allowing basic command-line interactions.
- A dynamic file explorer sidebar that mimics professional IDE navigation.
- Light/Dark mode toggling to enhance visual comfort.
- Session management through unique room codes that are easy to share and access.

CHAPTER-2

PROBLEM STATEMENT

2.1 Problem Statement

In today's fast-paced digital environment, the lack of a unified and accessible real-time collaborative development platform hinders productivity in remote teams, coding education, and technical assessments. Existing platforms are either complex to set up, restricted in functionality, or lack seamless multi-user interaction. There is a pressing need for a web-native solution that offers live coding, shared terminal access, role-based session control, and an IDE-like experience within the browser.

2.2 Motivation

The development of Collab Dev is motivated by several real-world challenges:

- Educators need an interactive and engaging tool for teaching programming where students can watch, collaborate, and ask questions in real time.
- Interviewers conducting remote technical interviews require platforms where they can observe and interact with a candidate’s code directly.
- Developers working in pair or group programming setups across different locations require efficient tools to co-code and test without having to sync their local environments.
- Mentors and learners benefit from instant feedback during code reviews or learning sessions.
- The success of collaborative text editors like Google Docs shows the viability and demand for similar tools in the software development space.

Hence, building a simple yet powerful platform that brings real-time collaboration to code editing becomes both timely and impactful.

2.3 Objectives

The primary objectives of the Collab Dev project are:

1. To design and implement a browser-based IDE for collaborative programming.
2. To enable seamless code synchronization and live updates between multiple participants.
3. To implement secure user authentication and role-based access control using Clerk.

4. To allow anonymous users to join as viewers using a unique room code.
5. To integrate a terminal emulator (Xterm.js) for basic in-browser command execution.
6. To simulate native IDE features like file explorers, code highlighting, and theme customization.
7. To ensure responsive design using Tailwind CSS and modern UI standards.
8. To provide session scalability and modularity for future feature extensions.
9. To serve multiple use cases such as coding bootcamps, technical interviews, academic coding labs, and collaborative development among remote teams.

CHAPTER-3

DETAILED SURVEY

3.1 CodeR: Real-time Code Editor Application for Collaborative Programming

The paper introduces CodeR, a real-time code editor application designed for collaborative programming. It aims to facilitate real-time collaboration among users working on the same project, supporting C, C++, and Java programming languages. The application provides a workspace for writing, executing, and displaying code results through a terminal, along with features like real-time collaboration, chat, and project management.

CodeR utilizes web socket technology to enable real-time collaboration, allowing multiple users to work on the same code simultaneously. It integrates with Facebook for user login and collaboration, and supports features like auto-saving, project file management, and terminal display for code execution results. The application also includes a chat feature for communication among collaborators.

The paper discusses the state of the art in real-time collaborative programming, highlighting the need for synchronization and real-time collaboration in software development. It mentions various tools and technologies that support real-time collaboration, such as Google Docs, EtherPad, and Ace. The paper also discusses the operational transformation algorithm used to maintain data consistency in real-time collaborative environments.

The application architecture is described, including user login, project file management, and collaboration features. The paper also discusses the evaluation of the application using Apache Benchmark, which measures the performance of the web server. The results show that the application can handle a large number of concurrent requests, with varying performance based on the computer specifications.

In conclusion, the paper presents CodeR as a web application that supports real-time collaboration in programming. It provides a workspace for code editing, execution, and project management, along with features like chat and terminal display. The application aims to improve the efficiency and quality of software development through real-time collaboration.

3.2 Real-Time Collaborative Coding in a Web IDE

This research introduces Collabode, a web-based collaborative integrated development environment (IDE) designed to support real-time, synchronous programming collaboration. Collabode enables multiple programmers to work on the same Java project concurrently, sharing edits instantly via a web browser. The motivation behind the system stems from limitations in traditional collaboration methods—either using a single shared IDE (limiting input to one user at a time) or managing separate versions with manual version control, which introduces delays and potential conflicts.

To address these issues, Collabode implements an error-aware integration algorithm that displays each programmer's edits in real-time while compiling changes individually. This ensures that developers see only the errors caused by their own code, allowing others to continue their work unaffected by incomplete or erroneous edits. Once a set of error-free edits is available, they are automatically integrated into the shared project. This mechanism balances collaboration and code integrity, enabling efficient team workflows.

The system architecture includes a client-side web editor using EtherPad for collaborative editing and a server-side backend that leverages Eclipse for compilation and execution. Collabode supports real-time feedback, automatic integration, and shared program output visibility. Three collaboration models explored in the paper—test-driven pair programming, micro-outsourcing, and mobile instruction—highlight the platform's flexibility for educational and professional use cases.

The paper evaluates the algorithm through simulations and a user study involving students and professionals working in pairs. Results showed that automatic integration occurred up to 2.8 times more frequently than manual commits would have, significantly improving collaboration fluidity. Participants praised the system's ability to isolate errors and support concurrent work, although some desired finer control over integration and more visibility into others' in-progress errors.

In conclusion, Collabode demonstrates the viability of a web-based collaborative IDE that maintains semantic correctness without the need for version control overhead, making it suitable for teaching, pair programming, and team development. The paper paves the way for more advanced collaboration models and highlights the growing importance of real-time, web-enabled development environments.

3.3 CoRED - Browser-based Collaborative Real-Time Editor for Java Web Applications

The paper introduces CoRED, a browser-based collaborative real-time code editor for Java applications. CoRED offers features like error checking, automatic code generation, and social media-inspired collaboration tools. It supports the Vaadin framework and can be used as a standalone editor or as part of other software, such as the Arvue IDE.

CoRED is designed to address communication issues in software development, especially in global software engineering. It leverages the web's collaborative capabilities and social media features to enhance communication among developers. The editor is built using the Vaadin framework, Ace editor, and Java Developer Kit (JDK).

The architecture of CoRED separates concerns between the client and server sides, with most processing done on the server. It uses differential synchronization for collaborative editing, allowing multiple users to edit the same document simultaneously. CoRED also includes features like code suggestions and error checking, which are generated on the server side.

CoRED can be customized and extended to support different frameworks and languages. It is modular, allowing for the replacement or extension of its components. The editor is planned to be published in the Vaadin Directory and will be part of the Arvue IDE, which is designed for creating and deploying Vaadin applications.

The paper concludes by highlighting the unique combination of features offered by CoRED compared to other browser-based code editors. It also discusses future work, including usability studies and extending the editor to support other phases of software development.

3.4 Connecting Programming Environments to Support Ad-Hoc Collaboration

The paper discusses the integration of collaboration streams into programming environments to support ad-hoc collaboration among distributed developers. It highlights the benefits of physical proximity in fostering opportunistic task adaptation and helping co-developers, and aims to replicate these benefits in a virtual setting.

The authors propose a collaboration-centered design that combines various collaboration streams, such as in-place text/chat, audio/video, real-time presence, and code sharing, into a single user

interface. This design is implemented using existing programming environment, communication, and compiler technologies.

The paper presents CollabVS, an extended version of Visual Studio, as a case study for this approach. CollabVS supports several new streams of interaction, including in-place audio/video, real-time presence at a granularity smaller than the file, and the ability to receive notifications when a developer has moved away from a program element of interest.

A user study with 16 participants showed that users liked the information in the various streams, found them easy to create and use, and were not bothered by privacy issues. The study also showed that each user used several streams simultaneously, and each stream was used by several users.

The paper concludes by discussing the potential applications of the collaboration streams, such as expertise identification, progress estimation, early code use and inspection, conflict detection and prevention, and opportunistic pair programming. It also suggests future work to extend the system to involve more than a pair of users and to investigate other collaboration streams and architectures.

3.5 Distributed Software Engineering in Collaborative Research Projects

The paper discusses the challenges and methodology of distributed software engineering in collaborative research projects. It highlights the need for effective alignment of development efforts with requirements from researchers and stakeholders, noting that these projects differ from commercial software projects.

The authors present a methodology that covers requirements engineering, architecture, issue tracking, and community building. This methodology is designed to be tailored to different project contexts and aims to support planning software engineering work in future projects.

The paper is based on experiences from two large-scale research projects, ROLE and Layers, funded by the European Commission. These projects faced common challenges such as decision making, short development cycles, and distributed teams.

The proposed methodology includes activities like technology surveys, requirements engineering, and technology assessment. It also emphasizes stakeholder engagement and the use of tools like the Requirements Bazaar and JIRA for issue tracking. The paper concludes by advocating for a professional culture of sharing and refining software engineering practices in research projects. It aims to provide a framework that can be adapted to the needs of future project consortia.

3.6 A Novel Approach for Collaborative Pair Programming

The paper presents a novel approach to collaborative pair programming in computer science education. The authors, Sanjay Goel and Vanshi Kathuria, propose a framework based on Dillenbourg's four conditions for collaborative learning. This approach aims to enhance problem-solving skills, improve work quality, and increase trust and teamwork among students.

The study was conducted with first-year engineering students in an introductory programming course. Students with no prior programming experience were paired, while experienced students worked solo. The experiment was well-monitored, with instructors and teaching assistants ensuring active participation and collaboration.

The results showed that inexperienced paired programmers performed at par with experienced solo programmers by the end of the semester. The paired students demonstrated better problem-solving skills, increased efficiency, and improved code quality. They also developed a stronger sense of trust and teamwork.

The authors conclude that their approach to collaborative pair programming is effective in helping inexperienced programmers learn quickly and perform well. They suggest that this method can be applied to other courses and recommend further studies to compare it with traditional pair programming and solo programming.

The paper also discusses the limitations of existing pair programming methods and highlights the need for a structured approach that ensures equal contribution from both students in a pair. The authors believe that their framework addresses these limitations and provides a more effective learning experience.

3.7 Syde: A Tool for Collaborative Software Development

Syde is a tool designed to enhance team awareness in collaborative software development. It addresses the challenge of maintaining awareness among distributed teams by sharing change and conflict information across developers' workspaces. The tool models source code changes as first-class entities, providing precise change information to interested developers.

Syde adopts a change-centric approach, treating changes as tree operations on abstract syntax trees (ASTs). This allows for more accurate change information and conflict detection by comparing tree operations. The tool provides real-time information on who is changing which parts of the system and detects merge conflicts as they arise.

Syde is an extensible client-server application with clients as Eclipse plug-ins. It captures changes at every save action and stores them in a centralized server. The tool provides various visualizations, such as decorations, word clouds, and buckets, to enhance workspace awareness and assist developers in collaboration.

The Conflict Plug-in displays information about emerging conflicts and provides a semi-automated process to resolve them. Syde has been used in various contexts, including industrial teams and student projects, and has been incrementally enhanced based on user feedback.

Syde is free and open software, available at http://syde.inf.usi.ch. It has been developed over two years as part of a Ph.D. thesis and is supported by the Swiss National Science Foundation. The tool aims to balance the tradeoff between providing relevant information and avoiding information overload.

3.8 Jimbo: A Collaborative IDE with Live Preview

Jimbo is a collaborative integrated development environment (IDE) designed to enhance team collaboration in software engineering and education. It supports real-time collaboration, communication, and live preview, making it beneficial for both professional and educational settings.

Jimbo's key features include synchronous and asynchronous collaboration, integrated communication tools like audio and text chat, and live preview for immediate feedback. These features help bridge communication gaps and foster mutual understanding among team members.

The IDE is web-based, requiring only a standard web browser, and offers a user-friendly interface consistent with popular IDEs. It uses an operational transformation algorithm to enable real-time code collaboration without conflicts.

Jimbo supports two main scenarios: educational settings, where it facilitates remote pair programming and live coding, and professional settings, where it integrates designers into the development process. This inclusion helps ensure that designs are accurately implemented and allows for immediate feedback.

The tool aims to improve today's CS education by bringing students closer to each other and their instructors, while also training them in collaborative practices consistent with current software engineering methods.

3.9 Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems

The paper titled "Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems" by Chengzheng Sun and collaborators addresses the fundamental challenges in developing collaborative editing systems that support real-time, distributed interaction among users. Such systems allow multiple users to concurrently view and edit shared documents, which can include text, graphics, or multimedia content, from different locations over a network. A core challenge in these systems is maintaining consistency across all user views of the document, especially when edits are made simultaneously or under conditions of network latency.

To tackle this, the authors propose a consistency model comprising three essential properties: convergence, causality preservation, and intention preservation. *Convergence* ensures that all users eventually see the same version of the document once all operations have been executed. *Causality preservation* guarantees that operations are executed in the order in which they were causally generated, preventing confusion from out-of-sequence actions. *Intention preservation* ensures that the effect of each user's operation is consistent with their original intent, even when concurrent operations from other users are involved.

To realize this model, the paper introduces a Generic Operation Transformation (GOT) control algorithm, which transforms operations before execution to ensure they do not interfere with one another. The approach includes inclusion and exclusion transformation functions that adjust operation contexts, maintaining each operation's semantic intent despite concurrent changes. These mechanisms allow collaborative systems to support unrestricted editing with minimal latency, without sacrificing consistency.

The paper also proposes a timestamping mechanism using state vectors to maintain causality, and an undo/do/redo scheme to ensure convergence when operations arrive out of order. A garbage collection algorithm is presented to manage the operation history buffer efficiently, allowing the system to discard operations that are no longer needed for transformation or recovery.

An Internet-based prototype was developed to test the proposed model and algorithms. The results demonstrate that the system successfully preserves consistency across distributed sites, supports real-time editing responsiveness, and handles complex concurrent editing scenarios. This work has laid the theoretical and practical foundations for many modern collaborative tools that use Operational Transformation (OT), such as Google Docs and Etherpad.

In conclusion, this research provides a robust and scalable solution for real-time collaborative editing, addressing key consistency challenges through a carefully designed combination of transformation algorithms, causality management, and system architecture.

3.10 Design and Development of Real-time Code Editor for Collaborative Programming

The article discusses the development of a real-time code editor, CrossCode, designed for collaborative programming. It highlights the shift from desktop to online platforms, emphasizing the need for tools that facilitate real-time collaboration without the hassle of lengthy configurations. The editor supports multiple programming languages and offers features like real-time chat, terminal building, and code execution.

CrossCode employs web socket technology to enable real-time collaboration, allowing multiple users to work on the same project simultaneously. It includes functionalities such as auto-saving, project management, and the ability to invite collaborators. The system uses algorithms like ReactJS for user interfaces and operational transformation for data consistency.

The implementation of CrossCode involves a login system, project management, and real-time collaboration features. Users can invite friends to collaborate, and the system supports real-time chat and code execution. The application is designed to enhance project development efficiency and facilitate remote teamwork.

The article also discusses the performance evaluation of CrossCode using Apache Benchmark, which measures the application's load time and responsiveness. The results provide insights into the application's effectiveness and scalability, highlighting the importance of continuous assessment and optimization.

In conclusion, CrossCode represents a significant advancement in online code collaboration tools. Future developments may include the ability to generate UML designs from code and support for multiple files within a single project. The goal is to evolve CrossCode into a comprehensive platform for managing both completed and in-progress projects.

3.11 V-Code: Online Code Editor

V-Code is an online collaborative code editor designed for remote software development teams. It features real-time collaboration, code compilation and execution, and error suggestions using machine learning techniques. The system is built using web technologies such as Nuxt.js, Node.js, Python Flask, and Express.

V-Code aims to cater to coding enthusiasts of all levels, offering features like live code sharing, a playground for multiple programming languages, and an online community for coders. The live code sharing feature allows users to collaborate on code in real-time, which is especially useful for remote team collaboration.

The system aims to develop a web-based application that offers a collaborative editor, compiler, and execution environment for programming languages. The collaborative editor will support multiple programming languages and provide real-time feedback to users on any syntax errors or other issues.

V-Code has three main modules: Live Code Share, Code Execution and Compilation, and Error Suggestion. The Live Code Share feature allows users to create a room and invite others to join and edit code in real-time. The Code Execution and Compilation feature allows users to execute code in various programming languages without having to download or install them on their own systems.

The Error Suggestion feature uses the OpenAI API to analyze the syntax of the code and recognize any potential errors or issues. By utilizing this API, V-Code can assist users in quickly detecting and correcting any problems with their code, ensuring that their programs operate smoothly and efficiently.

3.12 Collaborative Application Development Tool

The text discusses a research paper titled "Collaborative Application Development Tool" published in the European Chemical Bulletin in June 2023. The paper explores the evolution of software development from individual work to collaborative efforts, highlighting the importance of communication, coordination, and cooperation among team members. It emphasizes the need for integrated development environments that support real-time collaboration, code execution, and result display.

The paper reviews the history of collaborative development, from early version control systems to modern distributed systems like Git. It also discusses the challenges of manual merging and the

benefits of real-time collaborative coding. The research focuses on developing a web-based platform that enables real-time collaboration, chat, and project management, supporting languages like Python.

The proposed system includes features such as a code room for shared coding, a chat room for communication, and a blog space for idea sharing. It also supports online meetings and real-time error solving. The platform aims to enhance productivity and facilitate better application development through effective collaboration.

The implementation of the proposed system is described, emphasizing its web-based nature and real-time collaboration capabilities. The platform supports various programming languages and includes features like workspace provision, real-time collaboration, and terminal building. It aims to provide powerful tools for the development environment, leading to faster and more economic benefits for organizations.

The conclusion highlights the potential of the platform to revolutionize collaborative application development. Future work may involve integrating additional tools like Docker and supporting more programming languages. The research emphasizes the importance of real-time collaboration in improving software development processes and outcomes.

3.13 Enabling Seamless Software Collaboration: Design and Implementation of a Web-Based Collaborative Code Editor

The paper presents a web-based collaborative code editor designed to enhance productivity and teamwork among globally dispersed development teams. The platform features real-time code editing, an interactive whiteboard for brainstorming, and integrated chat for direct communication, supporting over ten programming languages. WebSockets enable low-latency, concurrent interactions, ensuring consistency and version control.

The system architecture includes a central server with a session manager and edit resolver, facilitating real-time editing and synchronization. Clients connect via WebSockets and HTTPS, with data stored in MongoDB. The platform aims to streamline collaborative workflows, reducing version conflicts and boosting efficiency.

The implementation involves setting up a development environment with Node.js and MongoDB, integrating a code editor like Code Mirror, and incorporating error highlighting tools. The front end supports basic file operations and real-time code execution, establishing a foundation for further enhancements.

The platform's outcomes include improved collaboration, real-time synchronization, and secure authentication. Future developments plan to enhance the interface with customizable themes, support for multiple languages, and integrated communication tools, aiming to create a more robust and user-friendly coding environment.

In conclusion, the collaborative code editor demonstrates significant potential to enhance software development for distributed teams. The platform's real-time synchronization, communication tools, and integrated coding environment facilitate smooth collaboration, with future improvements promising a more stable and adaptable environment for a broader range of users.

3.14 Online C/C++ Compiler using Cloud Computing

The paper presents an online C/C++ compiler leveraging cloud computing to address portability and storage issues. It allows users to compile code remotely, avoiding the need for local compiler installations. The system is platform-independent and stores errors/outputs conveniently, making it ideal for online examinations.

The online compiler consists of three main parts: the front end for syntax and semantics checking, the middle end for optimization, and the back end for language translation. It supports various compilers and provides a centralized repository for code, simplifying maintenance and distribution of dynamic usernames and passwords.

The project aims to provide a centralized compiling scheme for organizations and institutions, making user systems lightweight by eliminating the need for separate compilers/SDKs. It simplifies authentication and personalized task distribution, and ensures server safety by preventing malicious code execution.

The system uses a dual-layered architecture with clients on the lower layer and a server on the upper layer. The server handles client requests, compiles code, and stores client information. The 'cloud hard disk' is a shared resource, and the user interface is designed to be simple and platform-independent.

The online compiler is particularly useful for conducting university practical examinations. It allows students to log in with user IDs and passwords, start exams with a timer, and submit code for centralized evaluation. This eliminates the need for examiners to visit each machine and simplifies compiler upgrades.

3.15 Real-Time Collaborative Code Editor

The text discusses a research paper titled "Real-Time Collaborative Code Editor" published in the International Journal of Multidisciplinary Research in Science, Engineering, and Technology. The paper, authored by Dr. P. Manikandaprabhu and S. Shwetha, presents a web-based application developed using React.js, Node.js, and Socket.io. This application facilitates real-time collaboration among developers, offering features like syntax highlighting, automatic code formatting, version control, and error detection.

The paper highlights the importance of real-time collaboration in software development, breaking down geographical barriers and enabling seamless teamwork. It explores the functionalities and advantages of Real-Time Code Editor apps, emphasizing their role in streamlining the collaborative coding process and boosting productivity. The application discussed in the paper aims to revolutionize the software development industry by providing a dynamic platform for programmers to collaborate, code, and iterate in real time.

The text also mentions related works in the field of collaborative real-time code editors, citing studies that have investigated the structure and execution of such applications. It discusses the challenges and drawbacks of current real-time collaboration features in lightweight IDEs, highlighting issues like network latency, host dependency, limited language support, and debugging constraints.

The paper outlines the advantages of using a collaborative code editor, including understanding user needs, selecting the right technology stack, designing a robust architecture, implementing real-time communication, and developing conflict resolution mechanisms. It emphasizes the importance of version control, history tracking, testing, and quality assurance in ensuring the reliability and performance of the code editor.

In conclusion, the text underscores the transformative potential of Real-Time Code Editor web applications in facilitating remote collaboration among developers. It highlights the benefits of employing such tools and their prospects in reshaping the dynamics of collaboration in software development projects. The paper aims to provide a comprehensive insight into these applications and their significance in modern software development workflows.

3.16 Online Code Editor on Private Cloud Computing

The paper presents an Online Code Editor developed for programmers to write and modify code without platform requirements or specific physical computers. It runs on a Private Cloud using web technologies like HTML, PHP, CSS, and JavaScript. The editor features syntax highlighting, file management, and auto-save functions, utilizing the open-source software Ace for some functionalities.

The editor is deployed on a Private Cloud in the SaaS layer, accessible via various devices through an organization's intranet. It uses VMware vCenter Server for VM management and Appserv for server-side support. The architecture includes Server Side Engine and Client Side Engine communicating via HTTP(s), with Ace as the front-end editor.

Key functions of the editor include creating and managing projects and files, importing/exporting files, undo/redo, and running code. It supports HTML, PHP, CSS, and JavaScript. The editor was tested with multiple users, showing minimal resource usage increase.

The paper compares the Online Code Editor with Notepad++ and EditPlus, highlighting its additional features like project management and running code. It concludes that the editor is practical for Private Cloud use, allowing urgent code modifications without specific hardware or installations

Future work may include supporting more programming languages like C++ and C#. The editor aims to reduce organizational expenses on personal computers and provide a flexible coding environment.

3.17 An overview of platforms for cloud-based development

The paper provides an overview of state-of-the-art technologies for software development in cloud environments. It evaluates existing systems based on characteristics like applicability, productivity enhancement, support for collaboration, and post-development application hosting. The survey shows that while cloud-based development has made initial progress, significant challenges remain, such as integration of full capabilities, debugging, and runtime auditing support.

The paper discusses the evolution of cloud development tools from simple code editors to modern programming environments. It highlights the need for these environments to cover more than one stage in the development cycle, including programming, file version control, and deployment. However, the deployment of applications is still in early stages, requiring specialized cloud system administration skills.

The paper also reviews cloud software modeling, composition, and documentation tools. It notes that while some tools offer collaboration features, they often lack integration with other stages of the development process. The paper argues that a compact and reliable solution for cloud-based development does not yet exist, and that current efforts have focused on migrating desktop development technologies to the cloud.

The paper concludes that cloud development platforms have not yet provided a comprehensive solution for the development process as a whole. It suggests that a new programming model may be needed, one that focuses on the real problems of deployment, real-time updates, and resource monitoring. The paper also discusses the challenges of interoperability, security, and resource management in cloud-based development.

The text lists various cloud computing platforms, tools, and projects, such as Cloud4SOA, Cloud9, CloudForge, Codenvy, and others, along with their respective URLs. It also mentions several research papers and books related to cloud computing, software engineering, and data security. Additionally, it includes information about specific technologies like OAuth, OCCI, OpenID, OVF, and others, with their respective URLs. The text appears to be a compilation of resources and references related to cloud computing and software development.

3.18 Overleaf a collaborative cloud-based LaTeX editor

The text discusses a presentation titled "Overleaf: a collaborative cloud-based LaTeX editor" by Dr. Hamdane Mohamed Elkamel, presented at the Scientific Day & Workshop organized by the Applied Mathematics and Didactics Laboratory in December 2021. The presentation introduces Overleaf, an online LaTeX editor that allows users to write, edit, and publish scientific documents collaboratively.

Overleaf is highlighted as a solution to the problems faced with desktop LaTeX applications, such as setup, updates, and crashes. It offers a user-friendly interface, automatic saving of projects, and a wide range of templates for various document types. The platform also supports collaboration features, allowing multiple users to work on a document simultaneously and track changes in real-time.

The presentation provides a step-by-step guide on getting started with Overleaf, including registration, project management, and document compilation. It also mentions the integration of reference managers like Zotero and Mendeley, although this feature is only available to premium users.

Dr. Hamdane emphasizes the ease of use and accessibility of Overleaf, making LaTeX available anytime and anywhere. The platform also offers additional features like project review and submission as new templates. The presentation concludes with a summary of the benefits of using Overleaf for collaborative writing and document preparation.

The text also includes contact information for Dr. Hamdane and the Computer Science Lab at the Teachers' Training School of Constantine, Algeria. The presentation was uploaded to ResearchGate by Dr. Hamdane on December 13, 2021, and has been viewed 146 times.

CHAPTER-4

SURVEY SUMMARY TABLE

Sl no	Title	Problem	Method/Approach	Result
1	CodeR	Traditional IDEs lack real-time collaboration.	Web-based editor using WebSocket and user feedback.	Enabled real-time collaborative coding with useful features like chat & terminal.
2	Real-Time Collaborative Coding in Web IDE	Errors from collaborators disrupt coding.	Error-aware Java IDE (Collabode) that shares only error-free code.	Reduced code issues and improved code sharing.
3	CoRED	Web IDEs lack desktop-level features and real-time sync.	Feature-rich Java web editor with real-time tools.	Improved distributed team collaboration and coding efficiency.
4	Ad-Hoc Collaboration Support	Lack of spontaneous collaboration for distributed teams.	Enhanced Visual Studio with multiple collaboration tools.	Supported ad-hoc collaboration with benefits of co-located teamwork.
5	Distributed Software Engineering in Collaborative Research Projects	Distributed research teams face coordination and decision-making issues.	Combined EU FP7 methodology with open-source tools and stakeholder engagement.	Reusable methodology improving collaboration, tool integration, quality.
6	A Novel Approach for Collaborative Pair Programming	Traditional pair programming lacks true collaboration and teamwork skill development.	Students work independently first, then collaborate—based on a collaborative learning model.	Improved teamwork, problem-solving, and trust.

Sl no	Title	Problem	Method/Approach	Result
7	A Tool for Collaborative Software Development	Poor communication and delayed conflict detection in distributed teams.	Created Syde (Eclipse plugin) that tracks code changes in real-time using AST operations.	Detailed change alerts and conflict detection, improving coordination.
8	A Collaborative IDE with Live Preview	Real-time co-editing and designer-developer communication is hard remotely.	Developed Jimbo—web IDE with conflict-free collaboration, live preview, chat, and OT.	Enhanced real-time teamwork, especially for remote and design-involved development.
9	Achieving Convergence, Causality Preservation	Hard to keep consistency when multiple users edit together in real-time.	Proposed an algorithm ensuring consistency and built a prototype.	Solved major real-time editing issues effectively.
10	Design and Development of Real-time Code Editor	Desktop IDEs aren't great for team collaboration.	Built CrossCode using WebSockets with multi-language support and terminal.	Improved accessibility and remote collaboration.
11	V-Code: Online Collaborative Code Editor	Debugging together is inefficient with screen sharing.	Created V-Code with real-time editing and error suggestions using ML.	Improved collaboration and error handling efficiency.
12	Collaborative Application Development Tool	Manual code merging and slow teamwork reduce productivity.	Used web sockets for real-time editing, execution, and communication.	Cloud-based IDE that boosts productivity and reduces merge issues.

Sl no	Title	Problem	Method/Approach	Result
13	Design and Implementation of a Web-Based Collaborative Code Editor	Remote teams struggle with separate tools for coding, chat, and brainstorming.	Platform with real-time coding, chat, and whiteboard via WebSockets.	Unified workspace boosted productivity and reduced context switching.
14	Cloud-Based Online C/C++ Compiler	Lack of integrated tools for collaboration and synchronization in traditional IDEs.	Collaborative editor using React.js and Socket.io with syntax highlighting, version control, and shared editing.	Improved collaboration, code quality, and outperformed tools like Live Share.
15	Real-Time Collaborative Code Editor	Traditional editors lack real-time collaboration, hurting distributed development.	React.js and Socket.io-based editor with real-time features and error detection.	Enabled simultaneous editing, better code quality, and team productivity.
16	Private Cloud-Based Online Code Editor with ACE	Editing is hard without installed software, especially on the go.	Used ACE editor in private cloud to support web languages via browser.	Effective in portable scenarios, rivaling desktop editors in features.
17	Cloud-Based Development Platforms	Cloud IDEs lack full lifecycle features (debugging, modeling, deployment).	viewed tools like Cloud9, Codenvy, Codeanywhere.	No complete solution yet; collaboration & MDE are promising.
18	Overleaf	Desktop LaTeX editors are hard to manage and share.	Online LaTeX editor with real-time collaboration and templates.	Easy academic writing with live preview; limited in free version.

Table 4.1 Literature summary

CHAPTER-5

SYSTEM REQUIREMENT SPECIFICATION

5.1 Functional Requirements

Collab Dev is designed to provide a robust set of functionalities to support collaborative coding. The system enables users to create and join rooms, with authenticated users having the ability to create new rooms and both authenticated and anonymous users able to join existing rooms using a session code. Real-time code editing is facilitated through Socket.IO, ensuring that changes made by one user are synchronized across all participants in a room instantly, fostering seamless collaboration. Anonymous users are restricted to view-only mode, preventing them from editing code or executing terminal commands, which ensures that only authorized users can make changes. The Monaco Editor provides syntax highlighting and multi-language support for languages such as JavaScript, Python, and Java, enhancing the coding experience. Additionally, an inter active terminal powered by Xterm.js is available per room, allowing authenticated users to execute commands and view outputs in real-time, which is essential for debugging and running scripts collaboratively.

5.2 Non-functional Requirements

The non-functional requirements of Collab Dev focus on performance, security, usability, and scalability to ensure a reliable and user-friendly experience. Collaboration latency must not exceed 200 milliseconds for syncing code changes, ensuring that updates are reflected almost instantly across participants, while editor updates should occur within 100 milliseconds for sessions with up to 10 users, providing a responsive coding environment even under moderate load. Security is prioritized through Clerk, which secures authenticated routes, ensuring that only authorized users can edit code or execute terminal commands, and all communications are encrypted using HTTPS to protect data transmission. The user interface is designed to be responsive and intuitive, supporting theming with dark and light modes, and providing clear feedback for user actions to enhance usability. Scalability is addressed by enabling the system to support up to 100 concurrent users per room and handle multiple rooms simultaneously without performance degradation, ensuring that the platform can accommodate growing user bases. Reliability is maintained through Socket.IO reconnection mechanisms to handle network interruptions, and high availability is ensured with minimal downtime and regular data backups to prevent loss, making Collab Dev a robust and dependable platform for collaborative coding.

5.3 Hardware Requirements

The hardware requirements for Collab Dev are minimal, ensuring accessibility across a wide range of devices. The system operates on any standard device equipped with a modern web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge, requiring no specific hardware interfaces beyond standard browser capabilities, which makes it compatible with most laptops, desktops, and tablets. On the server side, sufficient resources including CPU, RAM, and storage are necessary to handle real-time communications and in-memory data management, particularly since Phase 1 of the system does not include database integration, relying instead on server memory to store session data and file structures, ensuring that the platform can support real-time collaboration without hardware-related bottlenecks.

5.4 Software Requirements

The software requirements for Collab Dev encompass a carefully curated stack of technologies designed to ensure robust real-time collaboration, a seamless user experience, and the potential for future scalability, addressing the needs of a collaborative coding platform. The backend relies on Node.js version 14 or higher as the runtime environment, utilizing its asynchronous, event-driven architecture to efficiently handle multiple concurrent users, while Express.js serves as the framework for managing server-side routing and API endpoints like /createRoom and /joinRoom, facilitating room management operations critical for collaborative sessions. Socket.IO is integrated to enable real-time, bidirectional communication, supporting events such as emitChange and broadcastToRoom to synchronize code edits and terminal outputs across participants with minimal latency, ensuring that collaborative sessions remain fluid and responsive even with multiple active users. On the frontend, Next.js version 12 or higher is employed, a React framework that supports server-side rendering and static site generation to enhance performance and SEO, while Tailwind CSS is used for styling, providing a responsive and visually consistent user interface across devices, ensuring components like the file explorer and editor adapt seamlessly to various screen sizes for an optimal user experience. Clerk is utilized for secure user authentication, managing sign-in processes, session tokens, and role-based access control, ensuring that only authenticated users can edit while anonymous users are restricted to view-only mode, although its UI customization requires wrapping default components to align with the system’s design aesthetic. The Monaco Editor powers the code editing functionality, offering syntax highlighting, auto-completion, and multi-language support for languages such as JavaScript, Python, and Java, providing a familiar and feature-rich environment that caters to diverse programming needs.

CHAPTER-6

SYSTEM DESIGN

6.1 System Design

6.1.1 System Architecture

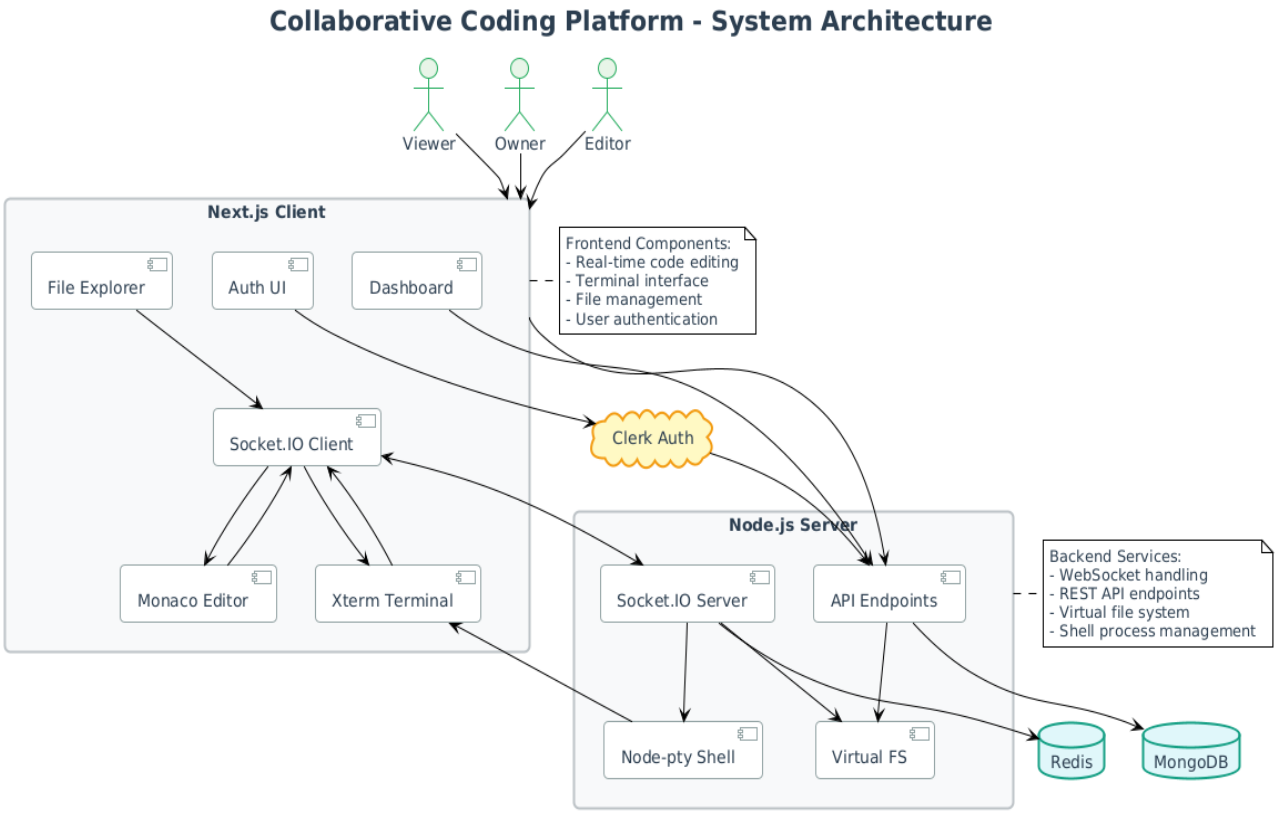

Figure 6.1 System Architecture

The system architecture of the Collab Dev collaborative coding platform is designed to enable real-time coding, file management, and terminal access through a structured interaction between the client, server, and external services. Users are categorized into three roles—Viewer, Owner, and Editor—with each role having specific permissions such as viewer_permissions, owner_permissions, and editor_permissions to control access levels. The Next.js Client serves as the user interface, incorporating a Socket.IO Client for real-time communication with the server via WebSocket, enabling events like code-update, file-select, terminal-input, join-room, and leave-room. Integrated within the client are the Monaco Editor for real-time code editing through code-change events, a File Explorer for file navigation and management via file-sync events, and an Xterm.js Terminal for command execution with terminal-input and terminal-output events. The client also includes an Authentication UI for user login through authenticate/sign-in and session management with

session_token, alongside a Dashboard/Room Page for displaying project rooms. The client communicates with the Node.js Server, which uses a Socket.IO Server to manage WebSocket connections and broadcast events like file-access, code-update, file-sync, and terminal-output to all users in a room, while also handling room management through create_room/file_ops. The server provides API Endpoints for operations like creating rooms and authentication, and integrates Clerk for user session and access control. External services include MongoDB for persisting project data and sessions, Redis for scaling socket rooms, Virtual FS for file operations like file-access and file-data, and Node-Pty for managing shell processes and spawning shells to handle terminal commands. Data flows through real-time collaboration as users connect to rooms via the client, sending events like code-change and terminal-input to the server, which broadcasts updates such as code-update and terminal-output to all room participants. File management occurs as the File Explorer sends file-select and file-sync events to the server, which interacts with the Virtual FS to retrieve or update files. Terminal commands are processed by the server using node-pty to spawn shell processes and return output. Authentication is secured through Clerk, validating credentials and issuing session tokens, while scalability is supported by Redis for socket room management, and persistence is ensured by MongoDB. This architecture enables Collab Dev to deliver a seamless, secure, and scalable collaborative coding environment with integrated code editing, file management, and terminal access features.

6.1.2 Module Design

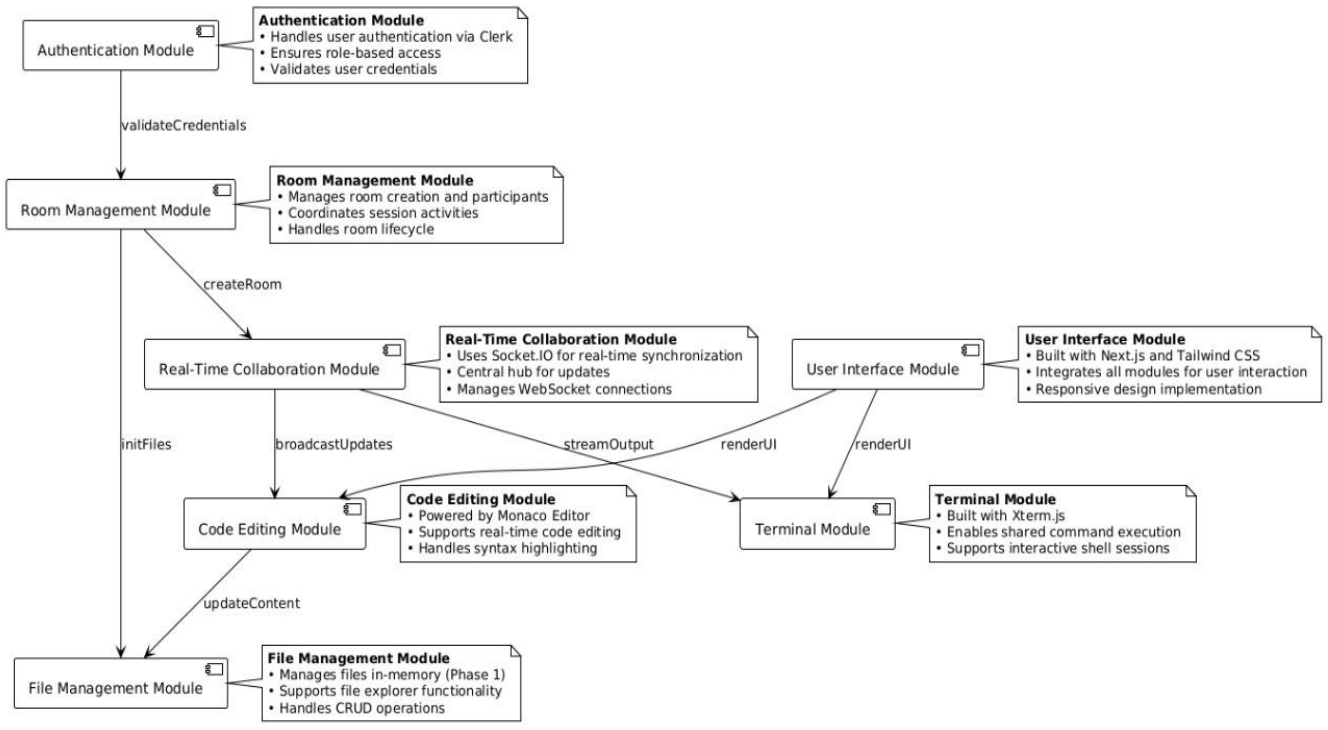

Figure 6.2 Module Design

The Collab Dev system is divided into seven modules for modularity and scalability. The Authentication Module uses Clerk for user sign-in, session management, and role-based permissions (Editor for authenticated users, Viewer for anonymous users). The Room Management Module handles room creation, joining, and in-memory state management, enforcing access control. The Real-Time Collaboration Module uses Socket.IO for low-latency (under 200 ms) code synchronization and terminal streaming, supporting events like emitChange and broadcastToRoom. The Code Editor Module integrates the Monaco Editor for multi-language editing with syntax highlighting and file explorer features. The Terminal Module uses xterm.js for an interactive terminal per room, restricting anonymous users to view-only mode. The UI Module, built with Next.js and Tailwind CSS, renders the editor, terminal, and supports theming (dark/light modes). The Error and Status Handling Module provides real-time feedback via Socket.IO. The design ensures separation of concerns, browser portability, and reliability with Socket.IO reconnection, meeting SRS requirements and enabling future scalability.

6.2 Detailed Design

6.2.1 Class Diagram

The Class Diagram for Collab Dev outlines the object-oriented design supporting real-time collaborative coding. It defines the system's major classes, their key attributes and methods, and the relationships that enable seamless collaboration among users.

The AuthManager class handles user authentication and access control. It manages the currentUser and provides methods such as validateToken for token verification, checkPermission for access control, generateInviteToken to create room invites, revokeToken to end sessions, and updateProfile for profile modifications.

The User class represents individuals in the system with attributes like id, username, email, avatarUrl, role, isAnonymous, createdAt, and lastActive. The method hasPermission checks if a user can perform specific actions. Users authenticate through the AuthManager to access collaborative features securely.

The SocketManager manages real-time communication. It maintains a connections map linking room IDs to WebSocket connections and includes methods like handleConnection to establish links, broadcastFromRoom to sync data among users, notifyUser to send updates, and clearUpDisconnected to manage inactive sessions.

The Room class defines collaborative workspaces. Each room has attributes like id, name, description, isPrivate, inviteCode, owner, createdAt, and lastUpdated. It provides methods such as addParticipant, removeParticipant, and changeVisibility. A room contains multiple EditorSessions, TerminalSessions, and Files.

The EditorSession class tracks live code editing within a room. Key attributes include id, roomId, currentFile, selections, cursors, and undoHistory. It supports methods like applyEdit to update code and syncCursors to maintain editing consistency among users.

The TerminalSession class handles shared terminal usage. It stores id, roomId, currentDirectory, commandHistory, and activeProcesses. Methods include executeCommand, clearHistory, and resizeTerminal, enabling interactive shell access in real time.

The File class models project files with attributes like id, name, path, content, type, language, createdBy, and createdAt. Methods like getFileTree, rename, and updateContent support structured and editable file systems within a room.

Class relationships ensure coherence: a User can join multiple Rooms; each Room owns several Files, EditorSessions, and TerminalSessions; the SocketManager syncs real-time actions. Together, these components deliver a cohesive and robust collaborative development platform.

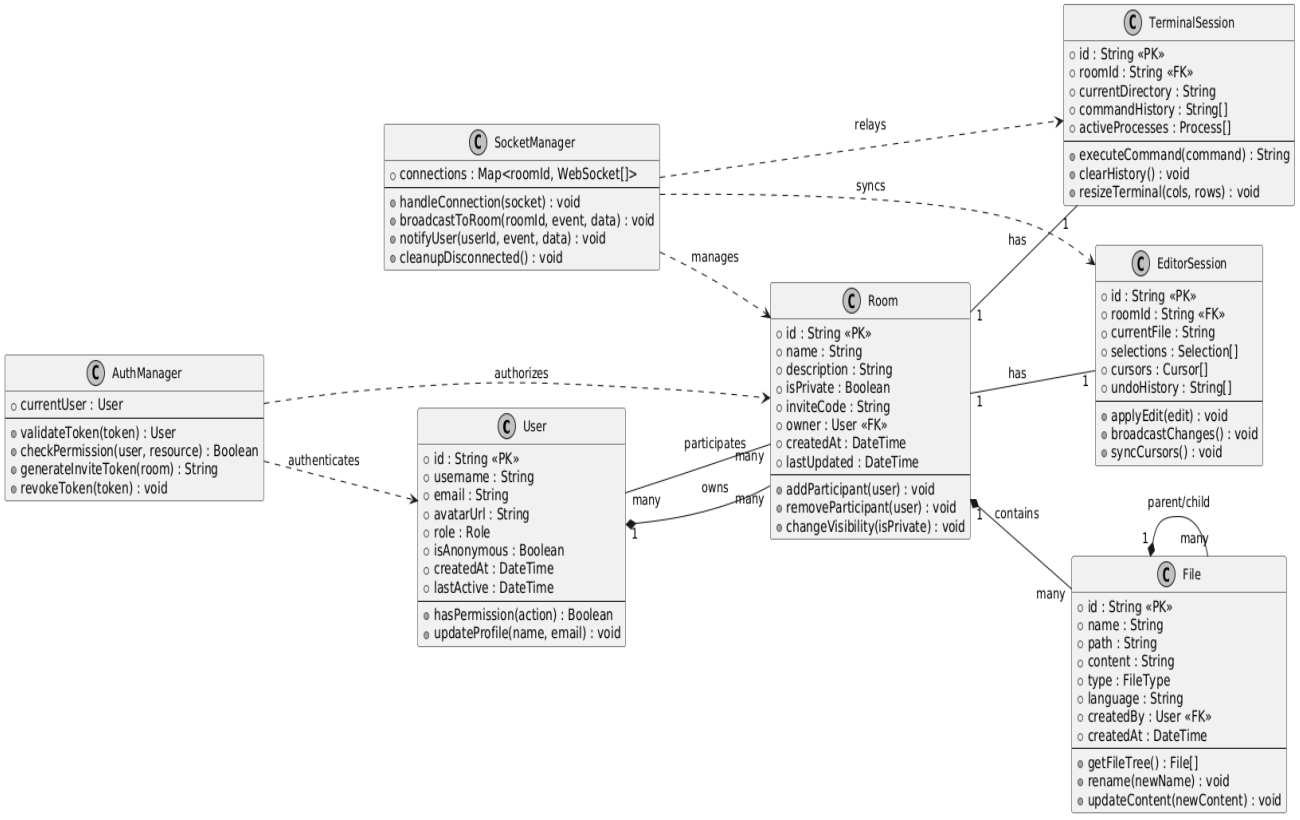

Figure 6.3 Class Diagram

6.2.2 Activity Diagram

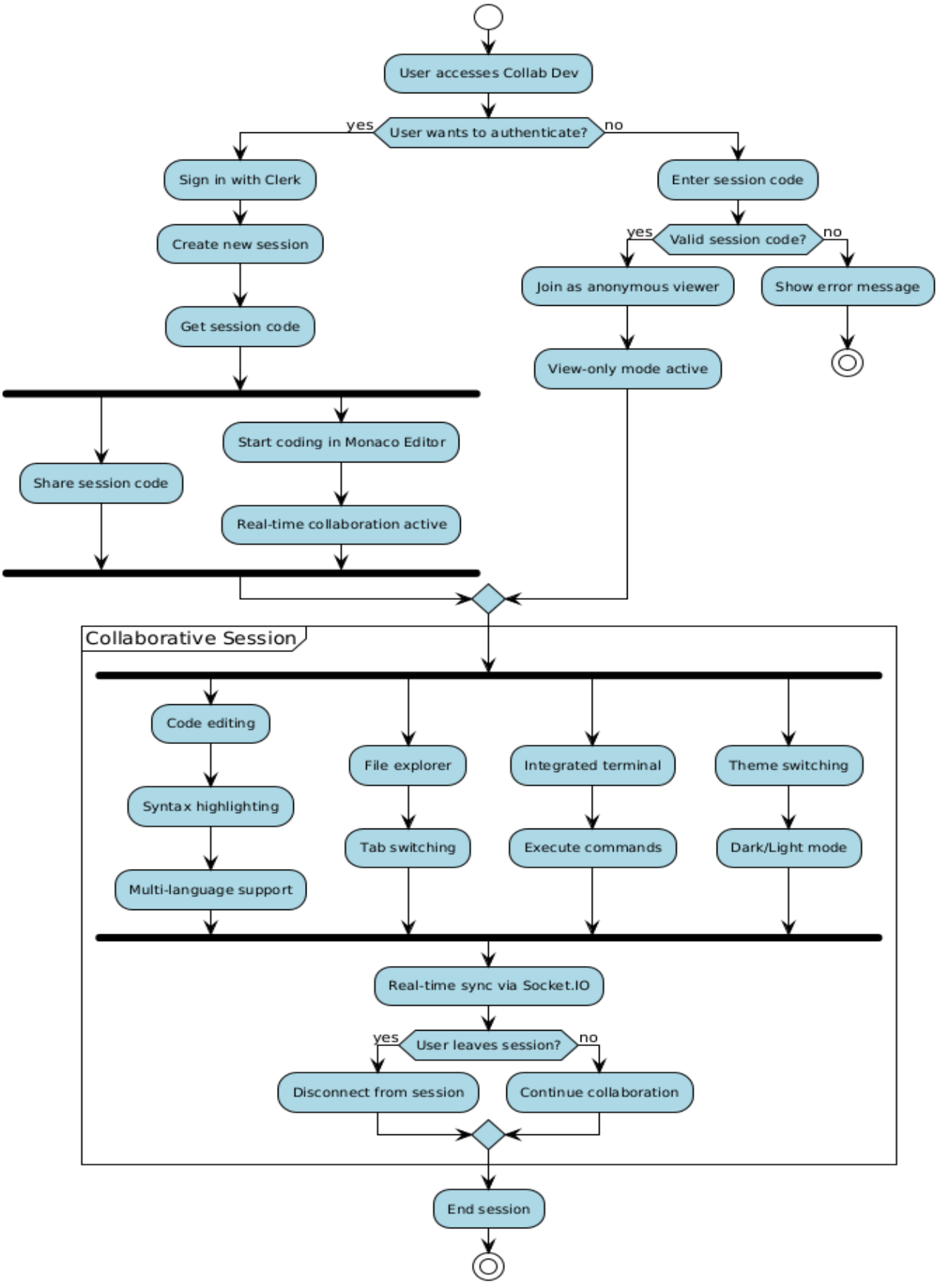

Figure 6.4 Activity Diagram

The Activity Diagram for Collab Dev illustrates the user workflow within the platform, capturing the sequence of actions, decision points, and real-time collaborative features that define a typical coding session. The process initiates when a user accesses the Collab Dev interface. At this point, the user must decide whether to authenticate using Clerk sign-in or join anonymously by providing a valid session code.

If the user opts to sign in, the system validates the credentials and grants access to the main dashboard, where the user can create a new coding session. This action generates a unique session code, which can be shared with others to enable collaboration. Alternatively, anonymous users can attempt to join a session using a code. If the code is invalid, they receive an error message and are denied access. If valid, anonymous users are typically granted view-only access to preserve session integrity.

Once inside a session, the system launches the Monaco Editor, enabling the user to begin real-time code editing. At this stage, collaboration features are activated, leveraging Socket.IO to synchronize code changes, cursor movements, and session states across all participants. The user can then share the session code to allow others to join the collaborative session seamlessly.

Within the session, users can perform various tasks: writing and editing code with syntax highlighting and multi-language support, navigating project files via a file explorer with tab switching, using an integrated terminal to run commands, and toggling between light and dark themes for a customized interface. All these features are kept in sync across users to provide a fluid collaborative experience.

The session persists until the user decides to leave. Upon exit, the user can either disconnect—ending their participation—or remain active to continue collaborating. This structured activity flow supports an intuitive, flexible, and real-time coding experience, central to the design goals of the Collab Dev platform.

6.2.3 Use Case Diagram

The Use Case Diagram for Collab Dev illustrates interactions between actors and the system, capturing role-based access, real-time collaboration, and session management workflows. Actors include the Authenticated User, who signs in via Clerk to gain Editor access, creating rooms, joining rooms, editing code, executing terminal commands, switching tabs, and changing themes; the Anonymous User, who joins rooms using a session code with Viewer access, viewing code and terminal output, switching tabs, and changing themes; the System Admin, managing sessions and assigning roles; and Clerk, the external authentication service. Use cases within the Collab Dev System encompass Sign In, handled by Clerk; Create Room and Join Room, both including Sign In

for authenticated users; Edit Code, extending to Execute Terminal Commands for authenticated users; View Code for anonymous users; and additional functionalities like Switch Tabs, Change Theme, and Manage Session for admins. Associations reflect user roles, with Authenticated Users accessing editing features, Anonymous Users limited to viewing, and Admins overseeing session management, ensuring secure and collaborative coding as per the SRS requirements.

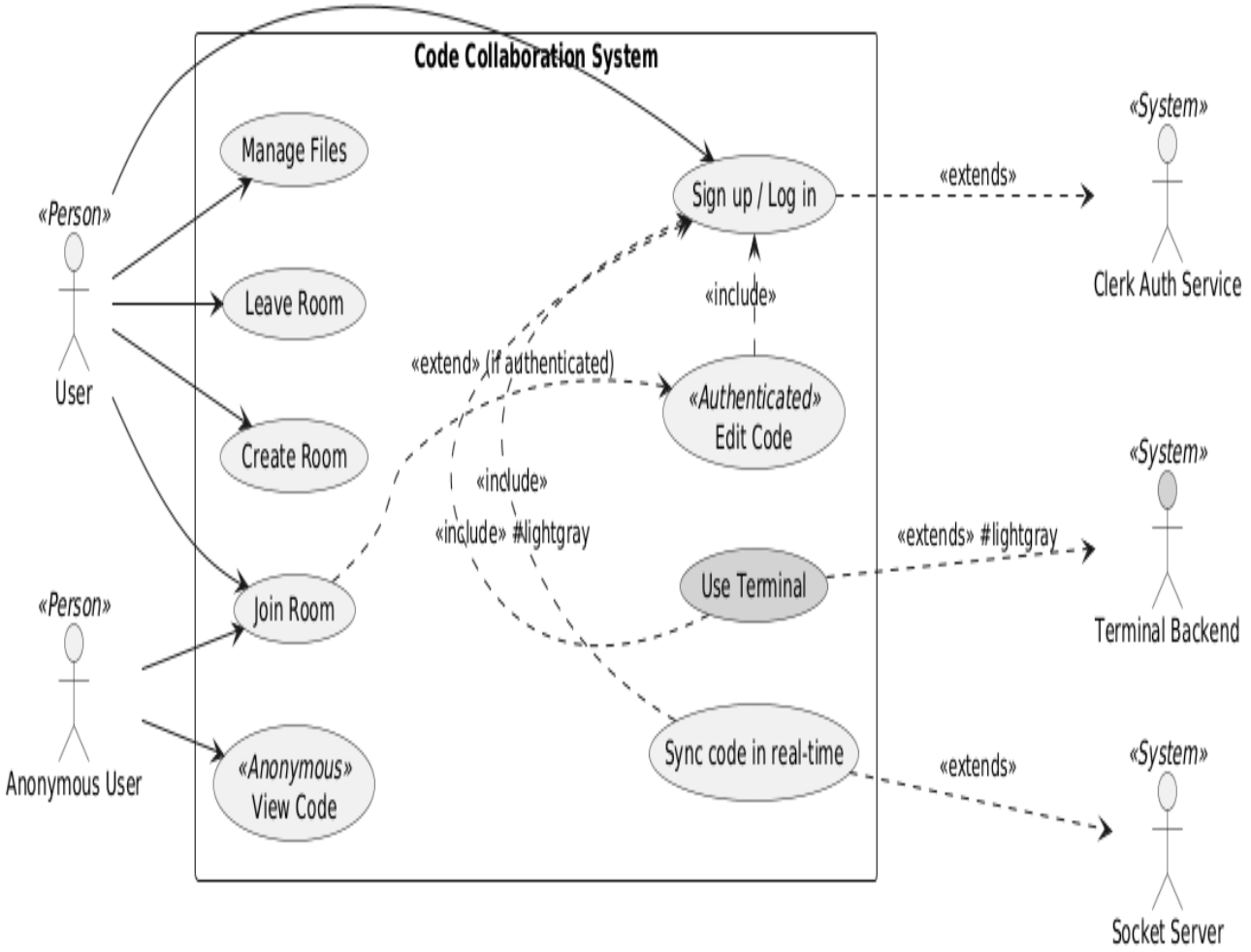

Figure 6.5 Use Case Diagram

6.2.4 Scenarios

The Sequence Diagram for Collab Dev captures the dynamic interactions between key entities—Authenticated User, Anonymous User, Room Manager, Socket Server, Code Editor, and File Storage—across the primary workflows of room creation, joining, and real-time collaboration. It illustrates the message exchanges and responsibilities of each component in delivering a synchronized coding environment.

The process begins when an Authenticated User initiates a session by sending a generateRoomID request to the Room Manager. In response, the Room Manager creates a default project file (e.g., JavaScript) through a createFile request to File Storage. Once successful, the Room Manager responds with a confirmation message including a RoomID (e.g., ABC_123). The user receives the RoomID and proceeds to load the session by triggering a loadRoom request, prompting File Storage to return the associated file content. This data is rendered in the Code Editor, initiating the collaborative coding interface.

During a live coding session, the Authenticated User performs edits in the Code Editor, which emits changes (emitChange) along with delta information to the Socket Server. The Socket Server broadcasts the updates to all room participants via a message containing the RoomID and delta. These updates are applied across all user instances of the Code Editor, maintaining real-time synchronization.

To join an existing room, an Authenticated User sends a joinRoom request with the RoomID to the Room Manager. Upon validating the RoomID, the Room Manager replies with a validRoom response and instructs the client to proceed with a loadRoomConfirmed message. File content is then retrieved from File Storage and rendered in the Code Editor, enabling full editing capabilities for authenticated users.

In contrast, an Anonymous User follows a similar joining flow by sending a joinRoom request with the RoomID. The Room Manager validates the code and, upon success, allows the room to load in read-only mode. The file content is retrieved from File Storage and rendered in the Code Editor, but without granting editing permissions—preserving the integrity of the collaborative session while still allowing anonymous access.

Overall, this sequence diagram highlights the message-driven architecture and clear role distribution among system components that ensure robust, real-time collaborative editing and secure room participation within the Collab Dev platform.

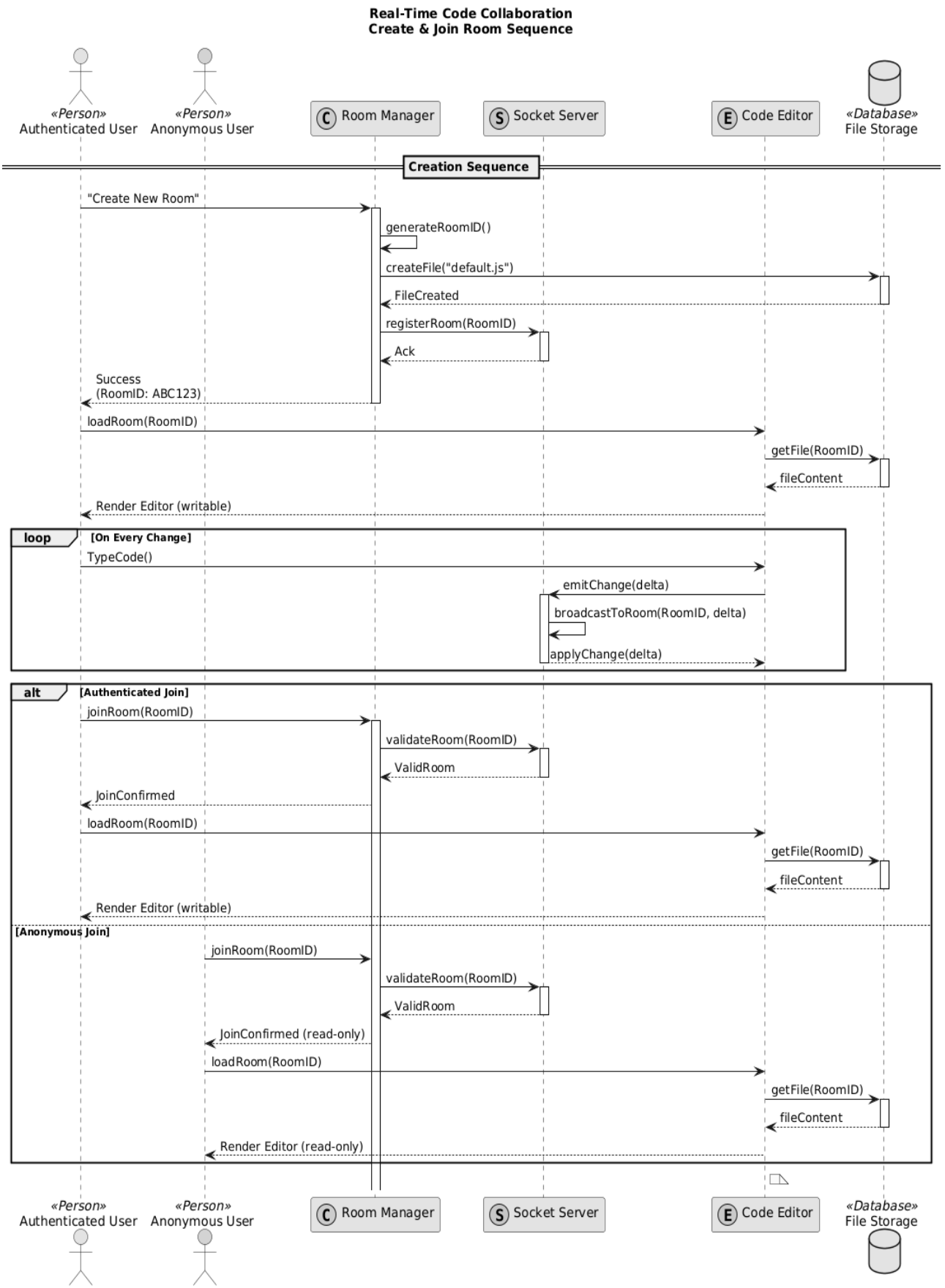

Figure 6.6 Sequence Diagram

CHAPTER-7

APPLICATION

Collaborative Coding Platforms

This system is ideal for platforms that aim to enable developers to collaborate in real-time, similar to tools like Replit or CodeSandbox. With integrated features such as live code synchronization, chat support, and role-based access, multiple developers can contribute to a single codebase without conflict. It can be used for remote pair programming, team code reviews, or even collaborative debugging.

Online Education and Coding Bootcamps

The platform can serve as a virtual classroom where instructors demonstrate code in real-time and students observe or interact. Educators can use it for live teaching, interactive coding assignments, and collaborative learning. The viewer-only mode is useful for large lectures, allowing students to follow along without editing the codebase directly.

Technical Interviews and Assessments

It provides an excellent environment for conducting live technical interviews. Interviewers can observe candidates in real-time, evaluate their thought process, and interact using an integrated terminal or chat system. The role-based access ensures that candidates cannot alter predefined environments.

Hackathons and Code Jams

Participants in coding contests, hackathons, or innovation challenges can use the platform to form teams and develop solutions collaboratively. With version tracking and file explorer features, managing contributions and maintaining project structure becomes efficient and intuitive.

Corporate Teams and Agile Development

Remote teams working in an agile environment can leverage this platform for sprint-based collaborative development. It facilitates real-time planning, brainstorming, prototyping, and deployment, eliminating delays associated with context switching between code editors, version control, and communication tools.

CONCLUSION

The development of Collab Dev marks a substantial advancement in overcoming the limitations of traditional, desktop-based coding environments. By delivering a seamless, browser-based collaborative platform, Collab Dev enhances accessibility and fosters real-time teamwork in software development. The platform integrates a range of modern web technologies, including the Monaco Editor for code editing, Socket.IO for real-time communication, Clerk for secure authentication, and a modular architecture that supports terminal interaction and dynamic synchronization.

This cohesive integration results in a user experience that closely mirrors native IDEs, while offering added benefits such as ease of access, cross-platform support, and minimal setup. Collab Dev effectively supports multiple real-world use cases—ranging from educational coding sessions and technical interviews to remote team collaboration and pair programming—highlighting its versatility and practical utility.

In comparison to existing tools and platforms, Collab Dev adopts and enhances best practices from the literature and industry, offering a unified, scalable solution with an intuitive interface and robust access control. Its design philosophy emphasizes simplicity, responsiveness, and flexibility, ensuring that both novice learners and experienced developers can effectively utilize the platform.

With the rising demand for real-time collaborative tools in globally distributed environments, Collab Dev establishes a solid foundation for future growth. Planned enhancements include multi-language support, version control integration, and AI-powered code assistance, all of which are aligned with the evolving expectations of modern development workflows.

In summary, Collab Dev emerges as a compelling and forward-looking solution tailored to the needs of contemporary software development and collaborative learning. It not only addresses current gaps but also paves the way for future innovations in the field.

REFERENCES

[1] Aditya Kurniawan, Christine Soesanto, Joe Erik Carla Wijaya. “CodeR: Real-time Code Editor Application for Collaborative Programming”, 2015.

[2] Max Goldman, Greg Little, and Robert C. Miller. “Real-Time Collaborative Coding in a Web IDE”. 2011.

[3] Janne Lautamäki, Antti Nieminen, Johannes Koskinen, Timo Aho, and Tommi Mikkonen.Janne Lautamäki, Antti Nieminen, Johannes Koskinen, Timo Aho, and Tommi Mikkonen. 2012.

[4] Rajesh Hegde,Prasun Dewan. “Connecting Programming Environments to Support Ad-Hoc Collaboration”.2009

[5] Michael Derntl, Dominik Renzel, Petru Nicolaescu, István Koren, Ralf Klamma. “Distributed Software Engineering in Collaborative Research Projects”. 2015.

[6] Sanjay Goel,Vanshi Kathuria. “A Novel Approach for Collaborative Pair Programming”. 2010.

[7] Lile Hattori and Michele Lanza. “Syde: A Tool for Collaborative Software Development”, 2010.

[8] Soroush Ghorashi,Carlos Jensen. “Jimbo: A Collaborative IDE with Live Preview”, 2016.

[9] Chengzheng Sun, Xiaohua Jia, Yanchun Zhang, Yun Yang and David Chen. “Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems”.

[10] Soumya Mazumdar, Prof. (Dr.) Sayantani Das, Prof. (Dr.) Saurav Naskar, Shivam Chowdhury, Disha Haldar, Ahana Bhattacharjee, Anjan Das. “Design and Development of Real-time Code Editor for Collaborative Programming”, 2024.

[11] Prof. Poonam R. Pathak, Tejas V. Magade, Avishkar A. Vichare Shreyas I. Repale. “V-Code: Online Code Editor”, 2023.

[12] Prof. Prajakta Pawar, Prof. Yogesh Kadam, Tushar Rokade, Saurabh Bhalerao, Adnan Khan, Nikhil Kumar. 2023.

[13] Ritesh Borale, Omkar Gunjote, Sarthak Chede, Siddhesh Bhelke, Shivaji Thengil. “Enabling Seamless Software Collaboration: Design and Implementation of a Web-Based Collaborative Code Editor”, 2024.

[14] Aamir Nizam Ansari, Siddharth Patil, Arundhati Navada, Aditya Peshave, Venkatesh Borole. “Online C/C++ Compiler using Cloud Computing”,2011.

[15] DR.P. Manikandaprabhu, S.Shwetha. “Real-Time Collaborative Code Editor. *ScienceDirect”*, 2024.

[16] Warangkhana Kimpan, Theerasak Meebunrot, Busaya Sricharoen. “Online Code Editor on Private Cloud Computing”,2013.

[17] G. Fylaktopoulos, G. Goumas, M. Skolarikis, A. Sotiropoulos and I. Maglogiannis. “An overview of platforms for cloud-based development”. 2016.

[18] Dr. Mohamed Elkamel Hamdane. “Overleaf a collaborative cloud-based LaTeX editor”. 2021.

[19] Mark Doernhoefer. “Surfing the Net for Software Engineering Notes”. 2021.

[20] Michael Armbrust, Armando fox, Rean Griffith, Anthony D.Joseph, Randy Katz, Andy Konwinski, Gunho LEE, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. “A view of Cloud Computing”,2010.

[21] Anton Beloglazov and Rajkumar Buyya. “Energy Efficient Resource Management in Virtualized Cloud Data Centers”,2010.

[22] Peter Mell, Timothy Grance. “The NIST Definition of Cloud Computing”,2011.

[23] Claudia Lavinia Ignat, Moira Norrie, Gérald Oster. “Handling Conflicts through Multi-level Editing in Peer-to-peer Environments”,2007.

[24] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, Scott R. Klemmer. “Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code”, 2009.

[25] Mark Graham. “Cloud Collaboration: Peer-Production and the Engineering of the internet”,2011.

[26] Richard C. Waters. “Program Editors Should Not Abandon Text Oriented Commands”.

[27] Yogesh Simmhan, Catharine van Ingen, Girish Subramanian, Jie Li. “Bridging the Gap between Desktop and the Cloud for eScience Applications”, 2010.

[28] Kai Tang, Jian Ming Zhang, Chen Hua Feng. “Application Centric Lifecycle Framework in Cloud”,2011.