



Project Report On

Collaborative Code Tool

*Submitted in partial fulfillment of the requirements for the
award of the degree of*

Bachelor of Technology

in

Computer Science and Engineering

By

Eldho George (U2103083)

Eswanth Sunil Dutt(U2103087)

Joe Joseph(U2103113)

Mohammed Kaif (U2103140)

Under the guidance of

Ms.Liya Joseph

**Department of Computer Science and Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)**

Rajagiri Valley, Kakkanad, Kochi, 682039

April 2025

CERTIFICATE

*This is to certify that the project report entitled "**Collaborative Code Tool**" is a bonafide record of the work done by **Eldho George (U2103083)**, **Eswanth Sunil Dutt(U2103087)**, **Joe Joseph(U2103113)** and **Mohammed Kaif (U2103140)** submitted to the Rajagiri School of Engineering & Technology (RSET) (Autonomous) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in "Computer Science and Engineering" during the academic year 2021-2025.*

Ms.Liya Joseph
Assistant Professor
Dept.of.CSE
RSET

Ms. Anu Maria Joykutty
Assistant Professor
Dept.of.CSE
RSET

Dr. Preetha K G
Professor & HOD
Dept. of CSE
RSET

ACKNOWLEDGMENT

We wish to express our sincere gratitude towards **Rev. Dr. Jaison Paul Mulerikkal CMI**, Principal of RSET, and **Dr Preetha K G**, Head of the Department of "Computer Science and Engineering" for providing us with the opportunity to undertake our project, "Collaborative Code Tool".

We are highly indebted to our project coordinator, **Ms. Anu Maria Joykutty**, Assistant Professor, Department of Computer Science and Engineering, **Dr. Jisha G**, Associate Professor, Department of Computer Science and Engineering, and **Dr Sminu Izudheen**, Associate Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our project guide **Ms.Liya Joseph** for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Eldho George

Eswanth Sunil Dutt

Joe Joseph

Mohammed Kaif

Abstract

The Collaborative Code Tool is an implementation aid in the form of an internet-based application for teamwork, programming, and learning. Traditional tools do not work well in team-based coding since they are always very passive and not very efficient; hence, they do not take over the feature of being synchronous learning/programming tools. This CCT is intended to overcome such limitations due to a number of factors that are present in the CCT, including its own group of high-tech built-in multi-user file synchronization, live code execution, and AI-assisted environment capabilities. Yjs is used in development for collaborative editing, persistent file storage, and direct communication with other users by means of Web Real-Time Communications.

In CCT, there is an AI module, which is a virtual tutor that helps users. The rich set of functionalities provided to the user includes real-time suggestions, simplification of complexity, translation of languages, and error detection while coding. It is useful for both novice and advanced users by originating specific recommendations based on the domain, renaming for better recognition, and auto-refactoring, which produces optimized code automatically.

Apart from the built-in options for multiple file and directory support and advanced concurrency management, CCT is an efficient, scalable, and highly interactive solution to the problem of collaborative coding. Compared to existing tools, CCT demonstrates its possibilities by effectively serving lower educational and professional purposes, thus making it apt for learning and project development.

It is safe to authenticate itself, and the files are secure and encrypted from malicious users as a measure to maintain data privacy, thus placing CCT in an optimal position for other forms of coding collaboration in educational, professional, and remote team environments.

Contents

Acknowledgment	i
Abstract	ii
List of Abbreviations	vii
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 Background	1
1.2 Problem Definition	2
1.3 Scope and Motivation	2
1.3.1 Scope	2
1.3.2 Motivation	3
1.4 Project Objectives	4
1.5 Challenges	4
1.6 Assumptions	4
1.7 Societal / Industrial Relevance	5
1.8 Organization of the Report	5
1.9 Conclusion	6
2 Literature Survey	7
2.1 OWebSync: Seamless Synchronization of Distributed Web Clients	7
2.1.1 Purpose	7
2.1.2 Implementation	7
2.2 Yjs for Real-Time Data Synchronization	8
2.2.1 Implementation	8

2.2.2	y-websocket for Client-Client Communication	8
2.3	Monaco Editor Code Editing Interface	8
2.3.1	Implementation	9
2.4	Data Binding for Real-Time Text Synchronization	9
2.4.1	Purpose	9
2.4.2	Implementation	9
2.5	Code Smell Detection using Whale Optimization Algorithm	9
2.5.1	Data Preparation	10
2.5.2	Discretization of Data	10
2.5.3	Initial Population Generation (WOA Setup)	10
2.5.4	Fitness Evaluation using Fisher Criterion	11
2.5.5	Execution of WOA (Optimization Phases)	11
2.5.6	Classification Rule Extraction	11
2.5.7	Model Evaluation	12
2.5.8	Advantages	12
2.5.9	Disadvantages	12
2.6	Data security algorithm for the cloud computing	13
2.6.1	Introduction	13
2.6.2	Methodology	13
2.7	Wersync: Synchronized Social Viewing Platform	14
2.7.1	Introduction	14
2.7.2	Methodology	14
2.7.3	Advantages	14
2.7.4	Disadvantages	15
2.8	Summary	16
2.9	Gaps Identified	16

3 Requirements 18

3.1	Hardware and Software Requirements	18
3.1.1	Hardware Requirements	18
3.1.2	Software Requirements	18
3.2	Functional Requirements	19

4	System Design	20
4.1	System Architecture	20
4.2	Component Design	20
4.2.1	User Module	20
4.2.2	File Module	21
4.2.3	Collaboration Session Module	21
4.2.4	Execution Module	22
4.2.5	AI Assistance Module	22
4.3	DataFlow Diagram	23
4.4	Work Breakdown	23
4.5	Project Timeline	25
4.6	Key Deliverables	25
5	Results and Discussions	26
5.1	Overview	26
5.2	Results	26
5.2.1	Real-Time Collaboration Analysis	26
5.2.2	Code Execution Performance	26
5.2.3	Data Storage and Management Solutions	27
5.2.4	AI Chatbot Integration	27
5.3	Discussions	27
5.3.1	Comparative Analysis with Other Systems	27
5.3.2	Challenges Encountered	27
5.3.3	Future Enhancement	28
5.4	Outputs	29
5.5	Summary	32
6	Conclusions	33
	References	34
	Appendix A: Presentation	35
	Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes	49

List of Abbreviations

CCT-Collaborative Code Tool

WOA-Whale Optimization Algorithm

List of Figures

4.1	Architecture Diagram.	20
5.1	Login Screen.	29
5.2	Interface.	30
5.3	AI Assistance.	30
5.4	File module.	30
5.5	Code Execution.	31
5.6	Group Chat	32
6.1	CO-PO and CO-PSO Mapping	54

List of Tables

2.1	Summary of Different methods their advantages and disadvantages	16
-----	---	----

Chapter 1

Introduction

The implementation effort being proposed in the present work is called The Collaborative Code Tool (CCT) project which is undertaken with an intention to design and develop a highly efficient web-based application that enhances the ability of collaborative coding sessions and other learning techniques. It affords feedback and interaction in a way that is wholly real time based. Designed for education and practice environments, for which purposes it is called CCT, integration to work makes the users work on both code and real projects. It allows the user to work interactively, determines that operations are performed online and marks out error and code. This way, combining user functions and security access the program provides efficient, entertaining and well-protected collaboration on software development and the application-centered approach towards coding based on learning.

1.1 Background

The requirement for an interactive coding platform results from the following issues in the existing development setting. Most code editor software available on the market does not support multi-user functionality, which enables more than one user to work on a project at the same time. This leads to situations where developers cannot observe each other's changes right away, and if they fail to notice a change made by a colleague, it may result in version conflicts and slow down the development process. Additionally, many platforms lack an effective real-time code compilation and error detection aspect, leaving users unable to quickly identify mistakes and receive immediate error messages to resolve them promptly.

Another significant concern is the restricted interaction capabilities of modern educational platforms. These platforms often provide simple coding environments that lack

control and real-time presentation or integration features. This limitation makes programming learning less engaging and reduces the retention of information. Furthermore, role management is underdeveloped on most platforms, despite its importance in defining responsibilities in group projects and helping individuals clearly identify their contributions to the team.

These challenges are addressed by the CCT project, which features Yjs file synchronization and WebRTC real-time communication. By incorporating these technologies, along with an AI-based virtual tutor, the platform provides additional features such as language translation, simplification, and error correction. These features make coding sessions more effective, user-friendly, and conducive to a productive learning environment.

1.2 Problem Definition

Traditional coding environments are often limited to individual workspaces, making it difficult for teams to work together seamlessly on the same project. This lack of real-time collaboration hampers efficient code development and knowledge sharing. Many existing tools do not support real-time code compilation and feedback across multiple users, leading to delays in identifying and fixing errors, which slows down the development process. Managing code files, directories, and project structures across multiple collaborators can be challenging without a unified platform that allows for organized and synchronized file sharing.

This project aims to address these challenges by introducing a collaborative coding tool based on a web-based platform. The tool is associated with real-time functionality, improved interaction, and flexibility within a multifaceted role-based perspective, streamlining responsibilities and enhancing teamwork..

1.3 Scope and Motivation

1.3.1 Scope

- Its functionality is to support Collaboration through the use of the Collaborative Code Tool (CCT) in order to enhance characteristics of traditional coding environments by improving the communication, collaboration between students and their

professors and teachers, presenting a new form of education, gamification, and practical improved programming languages.

- Provides a web-based platform where users can:
 - Multiple users can edit Python files in real time.
 - Directly execute the code and see error messages instantly.
 - Use user role management, secure authentication and a code edit section similar to a multi-file and directory compiler to VSCode.
 - Focuses on the educational field, to increase enrollment and interest in programming.
- Incorporates user role management, secure authentication, and a code editor similar to VSCode with multi-file and directory support.
- Targets the educational sector, making programming more accessible and engaging for learners.
- Current project scope includes:
 - Supporting Python and a few other languages.
 - Limiting to 20-30 simultaneous users per room for optimal performance and security.

1.3.2 Motivation

- Existing coding tools often lack real-time collaboration features, hindering seamless teamwork on projects.
- Educational platforms commonly offer limited interactivity, without real-time code execution or instant feedback.
- CCT is motivated to:
 - Increase user engagement and skill development by providing a collaborative, real-time coding environment.

- Integrate AI-powered assistance to enhance learning through code suggestions, error detection, and language conversion.
- Create a unified, interactive platform that supports both individual and team-based coding education.

1.4 Project Objectives

1. The primary objective is to create an engaging, web-based development and learning environment that incorporates real-time synchronous code editing.
2. The new features include real time code execution along with the identification of mistakes that the students could have made and syntax coloring.
3. Support different users to ensure that many users' roles are well catered to for improved merged functionality.
4. Offer secure access to the user login and room accessing with a set of built-in options.
5. Allow files to be kept updated in real time and co-authored.
6. Provide related AI support to help translate the code, make it less complex, and correct its errors

1.5 Challenges

Real-time collaboration can experience lag or interruptions due to internet speed variability. Additionally, handling multiple users editing simultaneously poses concurrency and synchronization challenges, while ensuring data security remains critical. Protecting code files and chat messages from unauthorized access or leaks also possess a concern.

1.6 Assumptions

- Users are expected to be familiar with collaborative tools similar to VSCode.
- The platform assumes a stable internet connection to support real-time collaboration.

- The application is designed to function optimally on modern browsers such as Chrome, Firefox, and Edge.
- A maximum of 20-30 users can join a room simultaneously.
- Basic encryption will be used for securing collaborative chat and code exchanges.

1.7 Societal / Industrial Relevance

This project addresses the need for interactive, real-time collaborative coding environments. Such platforms have vast applications in both educational and industrial settings, where teamwork and knowledge sharing are crucial. In education, this tool can enhance learning experiences by enabling real-time feedback and interaction. In the industry, it supports agile development practices and remote collaboration, fostering productivity in distributed teams. Overall, the platform is designed to streamline collaborative efforts, making coding and project management more accessible and efficient.

1.8 Organization of the Report

The CCT project report is segregated into three key chapters.

Chapter 1 establishes the foundation of the project by exploring its background, defining the problem it aims to solve, outlining its scope, and explaining the motivation behind its development. It also discusses the objectives of the project, the challenges faced during implementation, key assumptions, and its significance in both social and industrial contexts.

Chapter 2 (Literature Survey) delves into various real-time collaborative methodologies that have been used in other coding collaboration tools, providing a comparative analysis to highlight innovations and improvements introduced in the CCT.

Chapter 3 (Module Division) breaks down the project into its core functional components, detailing the different modules responsible for real-time collaboration, code execution, and user management, and the other mentioned essential features. The modules include User module, Execution module, Collaboration session module, File module, and AI Assistance Module. This structured approach ensures a comprehensive understanding of the development, implementation, and impact of the project.

1.9 Conclusion

The Collaborative Code Tool (CCT) project effectively transcends the limitations of the conventional coding interfaces by bringing in a collaborative interface that runs in real-time through the web. Through the use of technologies such as Yjs for synchronization, WebRTC for communication, and AI-powered support, CCT greatly enhances collaboration, learning, and project development. It brings in a new and engaging process of coding, enabling multi-user collaboration, real-time error detection, and user role management. By giving a high level of importance to educational spaces and professional collaboration, CCT fills the gap between conventional coding interfaces and modern, real-time collaborative interfaces. This project holds a lot of potential in both educational and industrial settings, enabling efficiency, accessibility, and innovation in software development.

Chapter 2

Literature Survey

2.1 OWebSync: Seamless Synchronization of Distributed Web Clients

The strong web technologies used to make this vision work have been combining React, Yjs, Monaco Editor, and a WebSocket server to permit effortless and real-time editing from multiple users. The design of this module is done with three core functionalities in mind: managing user interfaces, real-time content synchronization, and persistent connectivity between the clients. For the functionality of the following, here are the tools and technologies used:[1]

2.1.1 Purpose

React is used to add life and interactivity in the front-end interface of the code editor; as such, integration with user interactions and management becomes easy. Because of its component architecture, React can efficiently handle the states of an editor that simplifies the integration of varied collaborative features.

2.1.2 Implementation

The main React component, CollaborativeEditor, is the container for the code editor itself and also encapsulates the lifecycle for all major collaborative functionalities. For real-time updates, it exploits React hooks such as `useEffect` and `useRef` while DOM references are utilized.

2.2 Yjs for Real-Time Data Synchronization

Yjs is a library that provides the foundation to create a conflict-free replicated data type. In simpler words, Yjs is for real-time synchronization of complex data structures such as text documents even where users are typing in parallel.

2.2.1 Implementation

Yjs does create a Y.Doc, which is a kind of shared data structure that keeps track of all the insertions, deletions and so on from all the user actions. A central place to interact will be Y.Text type, the shared text field within the Y.Doc. Whatever changes a user does to any text will be captured by the Yjs framework for smooth real-time synchronization with no data conflicts.

2.2.2 y-websocket for Client-Client Communication

y-websocket establishes a WebSocket to enable the synchronization of constant data among multiple clients during runtime. The connection is crucial in low-latency collaborations for immediate change broadcast and reception. A WebSocket server is created with y-websocket and will listen on a certain port number, say 1234. For each of the clients, there is a connect() to this server through a WebSocketProvider, so they can now receive changes immediately whenever another client edits the shared document. This connectivity is the building block of the collaborative experience—when all the clients are working on the same shared document.

2.3 Monaco Editor Code Editing Interface

The Monaco Editor, being an editor that powers Visual Studio Code, is selected with the view of offering rich code-editing functionality such as syntax highlighting, auto-complete, and error-checking functionalities. [2]

2.3.1 Implementation

The Monaco Editor is loaded using the monaco-editor/loader package in a way that keeps it well integrated within the environment of React. After loading, the editor is bound to the Y.Text type through the MonacoBinding class from the y-monaco package. This allows the rendering of shared document by Monaco, and creates a real-time update instead through Yjs, assuring every modification made in the text on one client will be immediately displayed on any other connected client.

2.4 Data Binding for Real-Time Text Synchronization

2.4.1 Purpose

y-monaco MonacoBinding connects the shared Yjs text document Y.Text to the Monaco Editor model. It would propagate each character typed or deleted by a user to editors of all other users at once.

2.4.2 Implementation

Monaco can render immediate changes without writing a single line of code for synchronization through binding the Y.Text document to the Monaco Editor model. The binding is initialized in the CollaborativeEditor component and automatically updated by Yjs's CRDT algorithms. It gives an almost seamless editing experience across the users with immediate appearance of the changes.

2.5 Code Smell Detection using Whale Optimization Algorithm

The methodology in this paper applies the Whale Optimization Algorithm (WOA) to detect code smells in software systems, focusing on nine specific code smells. The process is structured into several stages, each designed to enhance detection accuracy and optimize performance.[3]

2.5.1 Data Preparation

Software Metrics Collection

Data is collected from five Java projects (e.g., ArgoUML, Azure, and Gantt Project) using 12 software metrics such as Lines of Code (LOC), Number of Methods (NOM), Cyclomatic Complexity (CC), and Lack of Cohesion in Methods (LCOM). Each metric provides insight into possible code smells, like Large Class or Long Method.

Training and Testing Split

The data is divided into 80% training and 20% testing sets to ensure reliable performance evaluation.

2.5.2 Discretization of Data

Metric Binning

A binning technique is applied to divide the range of each metric into discrete segments or bins. This simplifies the search process, enabling the WOA to categorize continuous metric data into specific ranges, enhancing the alignment of code smell detection rules with actual data.

2.5.3 Initial Population Generation (WOA Setup)

Random Initialization

WOA begins with a randomly generated initial population of possible solutions, where each solution is a set of if-then rules that combine different metrics.

Rule-Based Structure

Each rule corresponds to one of the nine target code smells, and the conditions (metrics and ranges) determine whether a module exhibits a particular smell.

2.5.4 Fitness Evaluation using Fisher Criterion

Objective Function

WOA employs the Fisher criterion as a fitness function, which assesses the quality of each rule by maximizing the separation (or “distance”) between instances with and without a code smell, while minimizing the variability within each group.

Fitness Calculation

For each rule (or search agent), WOA computes a fitness score, with higher scores indicating rules that better distinguish between smelly and non-smelly code.

2.5.5 Execution of WOA (Optimization Phases)

Exploitation Phase

WOA mimics the hunting behavior of humpback whales by adjusting search agents (rules) to converge around the most promising solutions. This includes:

- **Encircling** Agents move closer to the best solutions.
- **Bubble-Net Feeding** Agents move in spiral paths around optimal solutions to refine rule accuracy further.

Exploration Phase

WOA performs a broader search by randomly updating the position of agents, preventing premature convergence on suboptimal rules and allowing the discovery of new rule configurations.

2.5.6 Classification Rule Extraction

Optimal Solution Selection

Once WOA converges, the best-performing rules are extracted. These rules define thresholds and conditions that accurately detect each type of code smell based on metric values.

Rules for Different Software Sizes

Separate sets of detection rules are created for medium and large systems, considering the metrics relevant to each size category.

2.5.7 Model Evaluation

Performance Metrics

Precision and recall are calculated using a confusion matrix. Precision measures how accurately the model identifies actual code smells, while recall assesses the proportion of true code smells correctly detected.

Comparison with Other Algorithms

The framework's performance is benchmarked against existing methods (e.g., Genetic Programming and Competitive Co-Evolutionary Algorithm), showing improvements in precision and recall.

2.5.8 Advantages

- **High Detection Accuracy:** The methodology achieves high precision (94.42%) and recall (93.4%), which improves upon previous approaches like genetic algorithms.
- **Scalability:** The framework works well with both medium and large projects by adjusting detection thresholds for different software sizes.
- **Rule-Based Detection:** Using if-then rules provides clear, interpretable logic for developers to identify specific code smells.

2.5.9 Disadvantages

- **Computational Intensity:** WOA is computationally demanding, especially during the initial population generation and iteration processes.
- **Dependence on Discretization:** The methodology relies on binning metrics, which can reduce accuracy if not carefully chosen.

- **Limited Code Smells:** Only nine code smells are detected, out of the 22 identified by Fowler, limiting the framework’s comprehensiveness.

2.6 Data security algorithm for the cloud computing

2.6.1 Introduction

This paper presents a novel encryption algorithm that enhances cloud data security through a two-layer encryption method combining logical-mathematical operations and genetic processes. The algorithm, based on DNA cryptography, simulates natural genetic processes like transcription and translation to provide robust, randomized encryption. It addresses the computational challenges of traditional encryption, optimizing security and performance in cloud computing environments.[4]

2.6.2 Methodology

Two-Layer Encryption

The first layer employs the logical operations XOR, XNOR using partitioned data blocks of data part. second layer mimics DNA functionality; it transforms binary values into DNA nucleotides and then back.

Key Generation

Dynamic keys are generated based on genetic techniques, enhancing randomness.

Decryption Process

A reverse of the encryption steps, ensuring that only authorized users can decrypt the data.

Advantages

- Improved data protection with help of bio-inspired cryptography..
- Automated and perhaps, more volatile key generation.
- Up to moderately better protection against brute force.

Disadvantages

- Higher computational complexity.
- Increased encryption and decryption time compared to simpler methods.

2.7 Wersync: Synchronized Social Viewing Platform

2.7.1 Introduction

For the distributed users the Wersync allows for synchronized social viewing with the help of the Wersync platform a connection that makes it possible for them to communicate and cooperate through media use experiences. It offers features such as inter-destination media synchronization (IDMS) and inter-stream synchronization for multimedia on-demand an improved experience in collaboration. Wersync was designed in order to solve some problems with, build on the best of the existing platforms and is likely to offer a similarly integrated experience across various devices and a network environments.

2.7.2 Methodology

Previous research and a reflection on existing products relevant to Wersync were used in this study to inform its design. the art, user studies and other people's opinions. With this approach, it was possible to determine the necessary options, including operation of synchronized media; user interface instruments; and versatility platforms. Wersync's architecture adopts web element into different layers to ensure scalability and cross compatibility, employing a central server type of integration for synchronization problems communication as well as interaction among the users.

2.7.3 Advantages

- **Enhanced Synchronization:** Optimal identification of the different media types to be stored and the synchronization between streams solutions that enable reduction of latency and hurdles to the social viewing process.
- **Cross-Platform Accessibility:** Web-based design in order to have ease of access on various connections of gadgets and working frameworks.

- **Customizable User Experience:** Also as other features of free role and turning the students into the teachers and the teachers into the students.

2.7.4 Disadvantages

- **Potential Latency Issues:** Potential Latency Issues: However, even where optimization has been conducted, variations on the network might remain. This still results into occasional synchronization problems in less structured situations.
- **Increased Complexity for Non-Technical Users:** Increased Complexity for Non-Technical Users: While the platform offers other sophisticated collaborative tools, some of the individuals may be struggling provided without technical support..

2.8 Summary

Paper	Advantages	Disadvantages
Genetic Algorithm for Cloud Security	Provides enhanced data security through bio-inspired encryption and dynamic key generation.	Higher computational complexity and increased encryption/decryption time compared to simpler methods.
Wersync Platform for Synchronized Social Viewing	Enables seamless synchronized social viewing with cross-platform accessibility and customizable user experience.	Potential latency issues in uncontrolled environments and complexity for non-technical users.
Whale Optimization Algorithm for Code Smell Detection	Achieves high detection accuracy and scalability with interpretable rule-based detection.	Computationally intensive and limited to detecting only nine out of 22 code smells.
Real-Time Collaborative Code Editor with Web Technologies	Facilitates real-time editing and synchronization using React, Yjs, and Monaco Editor for multiple users.	Reliance on WebSocket connections can lead to latency in high-traffic scenarios.

Table 2.1: Summary of Different methods their advantages and disadvantages

2.9 Gaps Identified

- High computational complexity, impacting efficiency.
- Increased encryption and decryption time, limiting use in time-sensitive applications.
- Latency management challenges in uncontrolled network environments.
- Advanced features may be overwhelming for non-technical users, impacting accessibility.

- Computationally intensive during initial population generation and iteration, which affects scalability.
- Limited to detecting nine code smells, potentially missing other critical code quality indicators.
- Resource-intensive setup, which may not scale efficiently with a large number of users.
- Limited conflict resolution for complex, concurrent edits on nested structures.

Chapter 3

Requirements

3.1 Hardware and Software Requirements

3.1.1 Hardware Requirements

- **Processor:** 1.6 GHz or faster.
- **RAM:** Atleast 1 GB.
- **Disk space:** Atleast 500 MB.
- **Display Resolution:** Atleast 1024x768.
- **Operating System:** Preferably atleast Windows 7 or above.
- **Internet Connectivity:** Atleast 1 MBPS speed to ensure smooth collaboration.

3.1.2 Software Requirements

- **Backend:** Node.js or Python-based Server.
- **Node.js:** Required for WebSockets and Supabase integration.
- **Express.js:** If using Node.js for API handling.
- **WebSocket library:** For handling real-time updates.
- **Python (Latest stable version):** Required for execution.

3.2 Functional Requirements

- **Real-time Collaboration** – Multiple users can edit Python code simultaneously.
- **File Management** – Users can create, edit, delete, and organize Python files.
- **Secure Code Execution** – Executes user-submitted Python code in an isolated environment.
- **User Authentication** – Secure login and session management using Supabase.
- **Live Updates** – Automatic syncing of changes across all active users.
- **Role Management** – Differentiation between editors and viewers.
- **Error Handling** – Displays execution errors and syntax highlights.
- **Persistent Storage** – Saves files and execution history in Supabase Storage.

Chapter 4

System Design

4.1 System Architecture

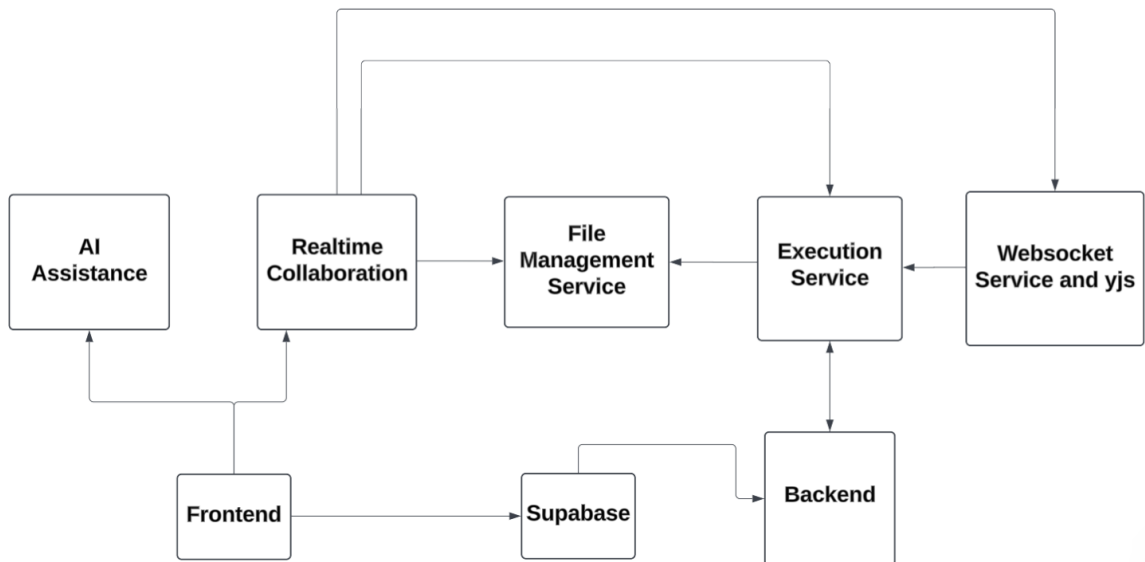


Figure 4.1: Architecture Diagram.

4.2 Component Design

4.2.1 User Module

The User Module is employed to depict each individual user on the platform, dealing with authentication, authorization, and user information. It ensures users are authenticated appropriately prior to entering a session so that security and access management can be guaranteed. For handling rooms, users must create a new room or join a current room via a room ID to provide structured collaboration. In addition, the module also keeps

user-specific data, like preferences, roles (e.g., admin or viewer), and session history, and provides a tailored and structured experience according to each user's needs.

The frontend is developed using React and integrates Monaco Editor to provide a seamless, VS Code-like coding experience within the browser. The system is designed to enable real-time collaboration by leveraging Supabase as the backend database and WebSocket provider. Additionally, the architecture allows for easy scalability and deployment, with Supabase handling authentication, storage, and database management, thereby simplifying backend complexities.

4.2.2 File Module

The File Module is made to show a Python code file that is simultaneously editable by multiple users at the same time. Along with Yjs, it also supports live syncing of file changes, so all the users get real-time updates. It further combines with a compiler interface to allow different users to execute code and observe results after a change, offering an interactive and seamless coding experience.

The file management system is implemented using local storage to allow users to create, edit, delete, and organize files within the frontend. The system is designed to mimic a lightweight file explorer, ensuring a seamless experience without requiring an immediate backend connection. When a user creates a new file or folder, the data is stored in local storage in a structured format. The React frontend retrieves and updates this data dynamically, ensuring persistence across sessions. Changes are saved back to local storage automatically, preventing data loss. This setup provides a responsive and efficient way to manage files on the frontend while maintaining flexibility for future backend integration.

4.2.3 Collaboration Session Module

The above Collaboration Session Module solely manages active collaborative editing sessions of Python code, enabling real-time collaboration among users. It employs Yjs to manage change synchronization, making it possible for the changes of multiple users to be merged harmoniously without data loss. Real-time communication is provided by Socket.io, which creates WebSocket connections to support efficient, low-latency data exchange among users. The module tracks session status—user activity and edited files—through a React-based user interface, which provides an immersive and dynamic user experience.

The user interface displays live updates such as cursors and selection highlights to represent the current editor. The inclusion of the Gemini API adds an AI-driven chatbot to the application, which provides smart code suggestions and assistance to facilitate collaboration. Updates are synchronized across all users in real-time using Yjs, where Local Storage is used for temporary management of files on the client side to provide fast access to session data. User information and session metadata are processed in Supabase, a scalable database created to manage authentication and long-term data storage. The module manages the entire lifecycle of the session—initialization and management of user entry and exit—up to maintaining the final document, thereby providing an end-to-end and integrated real-time collaborative code experience.

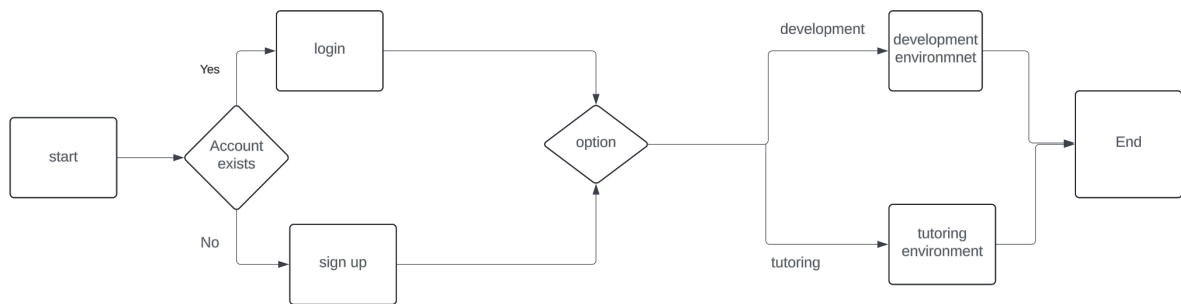
4.2.4 Execution Module

The Execution Module is used to handle compilation and execution of Python code within collaborative sessions. It provides real-time code execution, such that users can execute Python code collaboratively and see the results. For security and session integrity, the module runs in a secure and isolated environment, such that it does not allow interference between users. It receives output and runtime errors, showing them directly to users.

4.2.5 AI Assistance Module

The artificial intelligence processing module uses Google Generative AI (Gemini 1.5 Flash) to process Python code and user input to create meaningful responses. If a user provides a request in the form of a Python code snippet and some other text input, the backend system calls the AI model and provides a structured prompt that encompasses both aspects. The AI processes the received request and creates a suitable response, which is then extracted and formatted before it is returned to the user. If the AI cannot create a meaningful response, the system returns a fallback error message. Asynchronous management of API calls and response formatting is utilized to improve operational efficiency. This module helps the project by providing contextual feedback on the processing of Python code, thus making the development process interactive and informative..

4.3 DataFlow Diagram



4.4 Work Breakdown

Frontend Development (Team Member 1):

- User Module: Develop a simple-to-use interface using React for login, registration, and user profile management. Implementing real-time user status updates (online/offline status indicators) with Socket.io.
- Collaboration Session Module: Create a multi-user interactive code editor UI with React that can handle editing and live updates. Embed Yjs within the client-side UI to provide real-time presentation of synchronized modifications, cursor positions, and highlighted selections. Implement a chat functionality in the session user interface, utilizing Socket.io to facilitate real-time communication.

Backend Development (Team Member 2):

- User Component: Set up Supabase to cache user data, such as authentication information and session history. Add user login/logout, session creation, and join APIs for secure access control.
- File Module: Utilize Local Storage to store and manage temporary files on the client side. Integrate backend functionality to move files from Local Storage to Supabase and store them persistently across sessions. Implement APIs to handle file uploads, downloads, and directory hierarchy.
- Group chat: Implemented groupchat using supabase as database

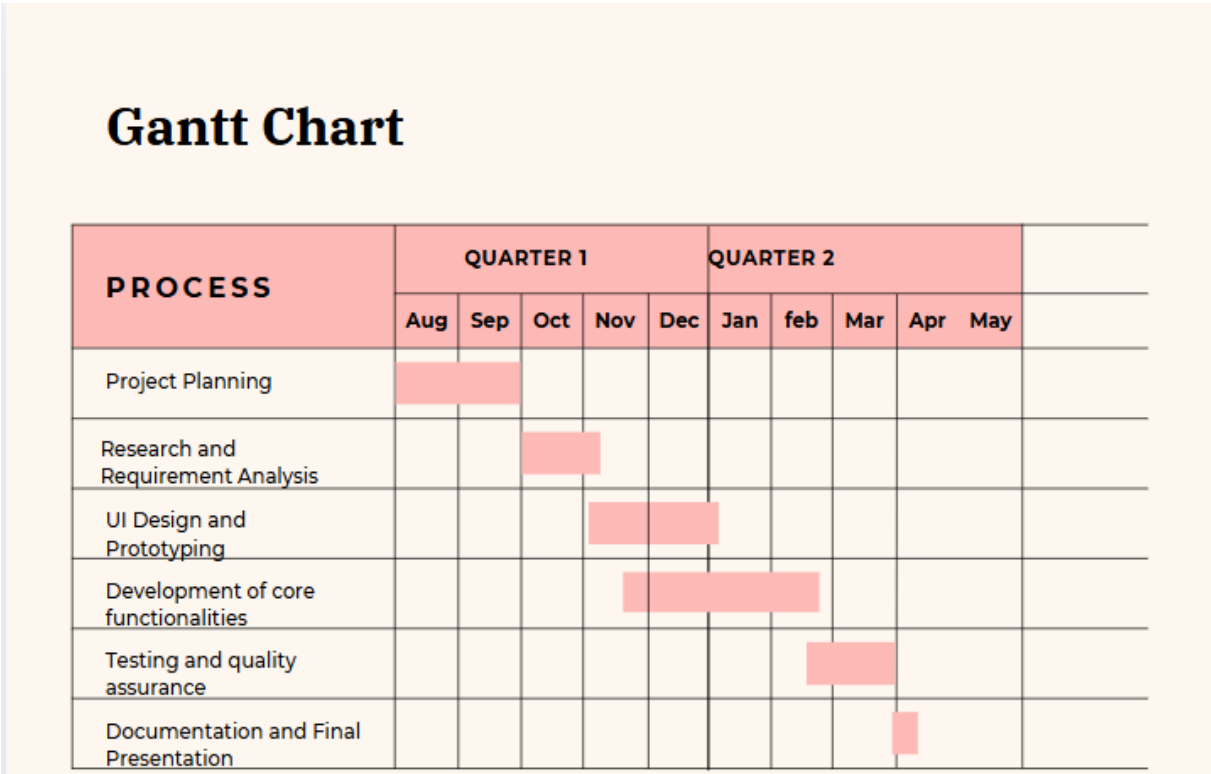
WebRTC / Socket.io Integration (Team Member 3):

- Collaboration Session Module: Use Socket.io to establish WebSocket connections for real-time data transfer and user communication. Manage session state and client multiplexing to enable smooth data exchange during simultaneous editing. Collaborate with Yjs to synchronize changes and offer consistency to all users involved.
- Execution Module: Use an execution engine for natively embedded code in the React interface that allows users to run Python (or any other language) files in real-time. Utilize Socket.io to send execution outcomes to all session participants in real-time.

AI Assistance (Team Member 4):

- AI Assistance Module: Integrate the Gemini API to provide a communication interface for an AI chatbot responding to code-based questions. Employ React to provide frontend functionality for showing AI-created responses, including coding suggestions and explanations for errors. Install features for language code translation, reducing complexity, and offer alternative options using the Gemini API.
- Execution Module: Save execution results to Local Storage for duration and sync with Supabase for session logs. Enable smooth communication among the execution engine, Yjs for collaborative updates, and the Gemini API for AI-enhanced debugging.

4.5 Project Timeline



4.6 Key Deliverables

When the website is visited the user will be able to view an interface where they must enter the room id and username using which they will be able to access a page where user will have facility to create files and write the program in it.Different individuals can write the code in same room and changes can be viewed by all members in real time.Now they will be able to compile the code and also can make use of the ai for any kind of help they need.

Chapter 5

Results and Discussions

5.1 Overview

This chapter outlines the results, discussions, and findings collected from the collaborative code compiler project. The primary objective of the project was to create a real-time and efficient collaborative coding environment using contemporary web technologies. The results are assessed in terms of real-time synchronization, efficiency of code execution, user experience, and system scalability.

5.2 Results

5.2.1 Real-Time Collaboration Analysis

The following broad areas were explored to establish the effectiveness of real-time collaboration:

- **Multi-user Synchronization:** Implemented with the help of Yjs, allowing users to edit code collaboratively in real-time.
- **Conflict Resolution:** Efficient handling of concurrent changes by means of Yjs' conflict-free replicated data types (CRDTs).
- **Latency Analysis:** Minimal latency in mirroring updates among users to facilitate smooth collaboration.

5.2.2 Code Execution Performance

- **Execution Speed:** The system executes code submissions with efficient speed, with little delay.

- **Error Handling:** Offers precise error messages for debugging, enhancing developer experience.
- **Multilingual Support:** Enables the running of various programming languages, thus making it more usable.

5.2.3 Data Storage and Management Solutions

- **Use of Local Storage:** Allows users to save and load code snippets effectively.
- **Database Performance:** Supabase performs user data management and session persistence efficiently.

5.2.4 AI Chatbot Integration

- **Accuracy and Responsiveness:** The Gemini API offers real-time coding suggestions and context-aware suggestions.
- **User Interaction:** Boosts productivity through the presence of AI-driven debugging and code enhancement.

5.3 Discussions

5.3.1 Comparative Analysis with Other Systems

- **Performance Benefit:** Unlike other collaborative coding tools, the combination of Yjs and Socket.io offers lower latency and better synchronization.
- **AI-Assisted Coding:** Unlike traditional compilers, the integration of the Gemini API makes the coding process more intelligent with intelligent suggestions and help.
- **Scalability:** Supabase and WebSockets usage allow better scalability in handling large numbers of simultaneous users.

5.3.2 Challenges Encountered

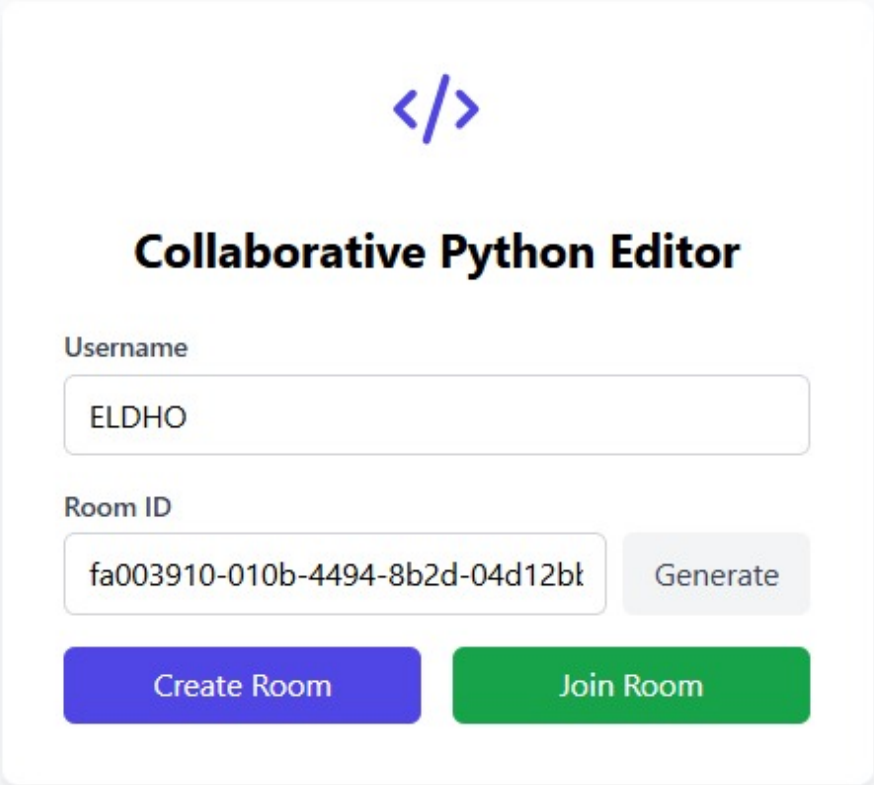
- **Synchronization Issues:** Early issues with multi-user editing were resolved by improving Yjs integration.

- **Heavy Resource Consumption:** Real-time updates and AI processing demand high computing resources.
- **Security Issues:** Safe data storage and transmission remain an issue.


5.3.3 Future Enhancement

- **AI Chatbot Optimization:** Enhancing the response time and the accuracy of suggestions.
- **Scaling Up:** Employing load balancing techniques to support more users.
- **Improved File Management:** Adding cloud storage capability for improved file accessibility.
- **Real-Time Debugging Tools:** Enabling collaborative debugging for users by leveraging AI-powered insights.
- **User Control:** Implementing user specified access control based on roles like viewer, owner .

5.4 Outputs



The image shows a login screen for a 'Collaborative Python Editor'. At the top, there is a purple icon of a code block with a slash and a greater-than sign. Below this is the title 'Collaborative Python Editor' in bold black text. The form contains two input fields: 'Username' with the value 'ELDHO' and 'Room ID' with the value 'fa003910-010b-4494-8b2d-04d12bt'. To the right of the 'Room ID' field is a 'Generate' button. At the bottom, there are two buttons: 'Create Room' in blue and 'Join Room' in green.



Collaborative Python Editor

Username

Room ID

Figure 5.1: Login Screen.

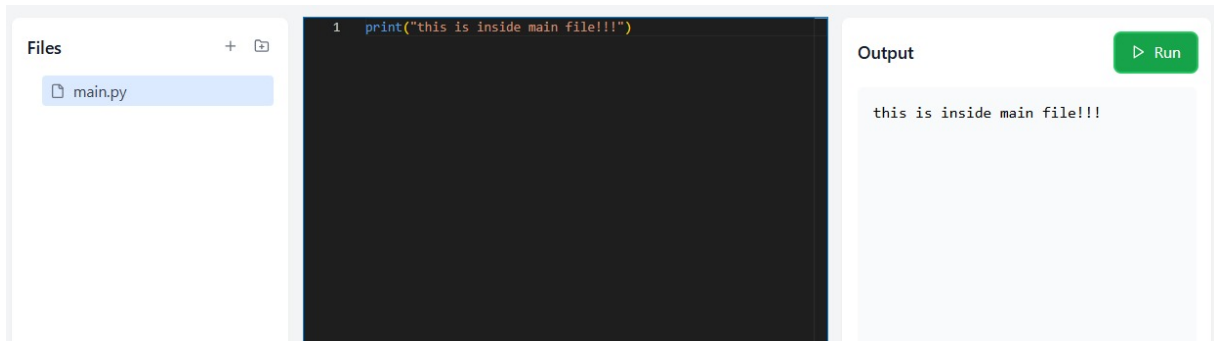


Figure 5.2: Interface.



Figure 5.3: AI Assistance.

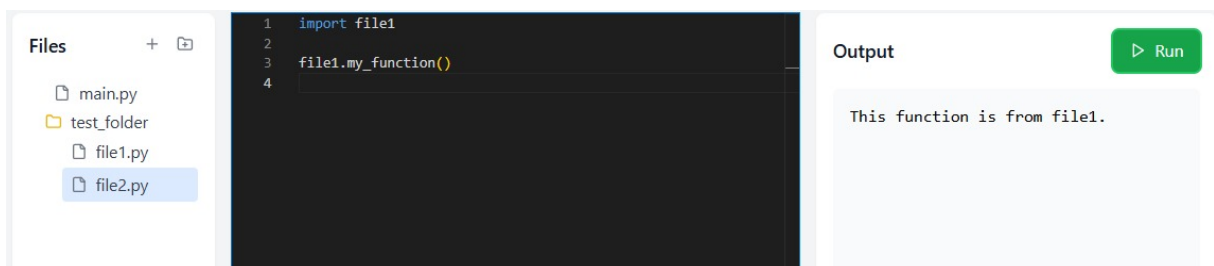


Figure 5.4: File module.

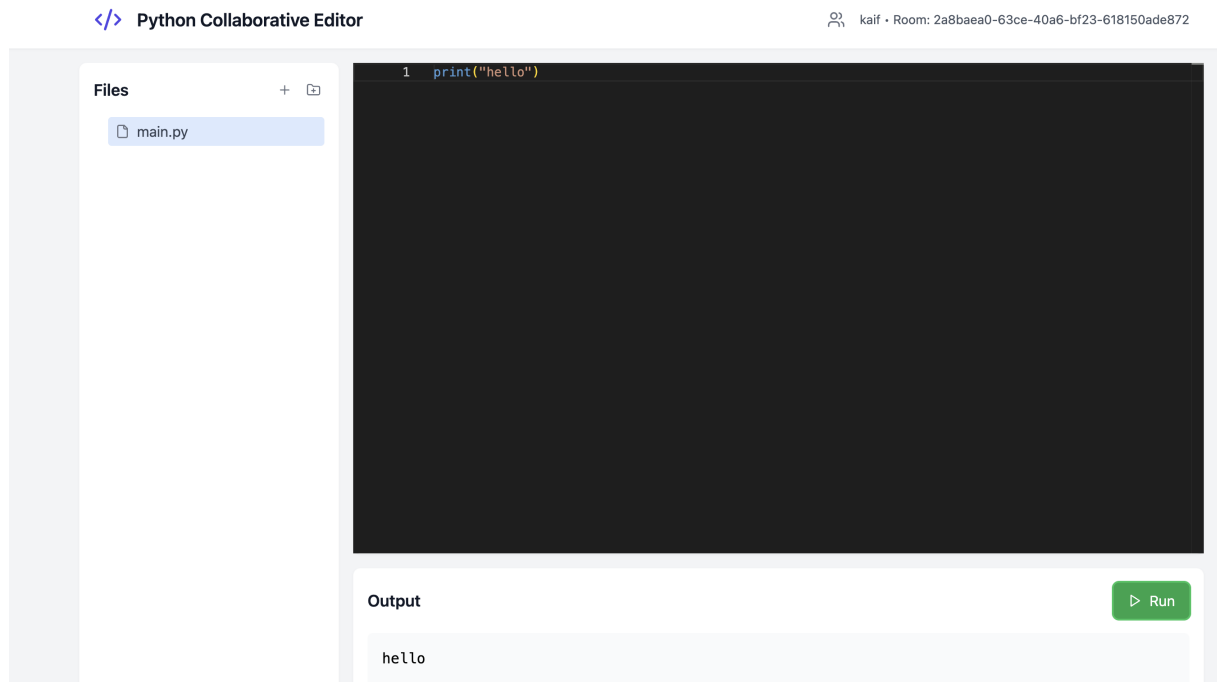


Figure 5.5: Code Execution.

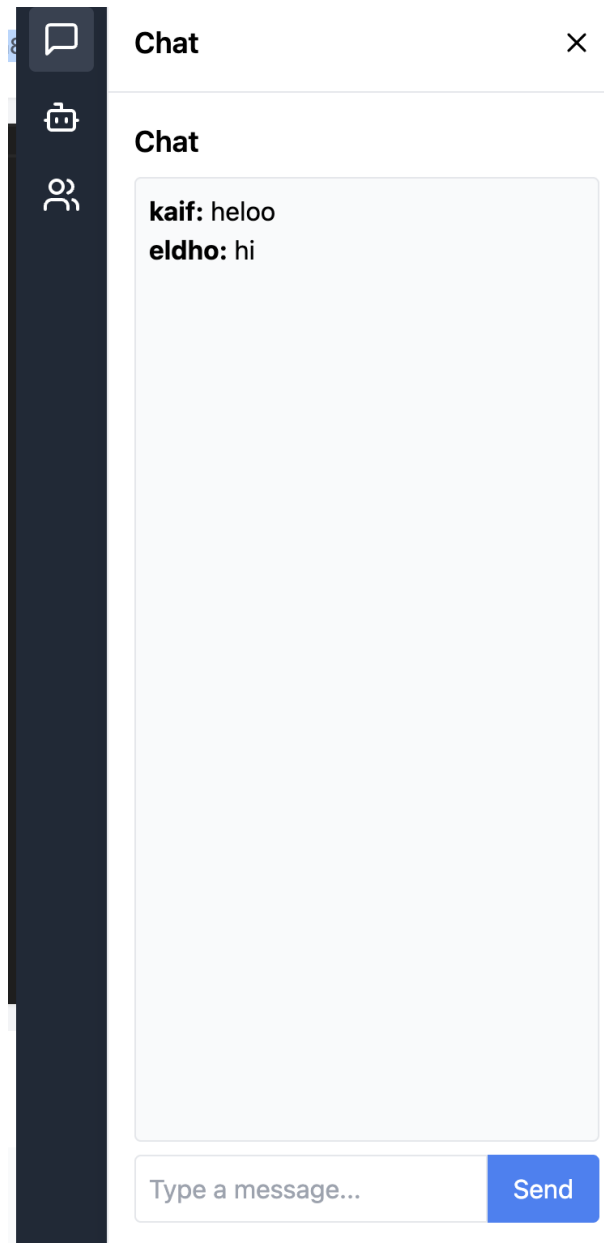


Figure 5.6: Group Chat

5.5 Summary

The outcomes of the cooperative code compiler project show the successful deployment of an AI-assisted, real-time coding system. The integration of Yjs, Socket.io, and the Gemini API significantly improves the user experience with seamless collaboration and smart help. The system shows promising applications in remote coding, pair programming, and online code testing despite some problems.

Chapter 6

Conclusions

The Collaborative Code Tool (CCT) successfully offers a real-time, interactive, and efficient coding platform using modern web technologies. The project supports effortless collaboration of users with Yjs, offering real-time synchronization of code changes without conflicts. The user interface is supported by React, offering a responsive and dynamic coding experience. Socket.io is a central feature that allows real-time communication and data transfer, enabling multiple users to collaborate in real-time without latency. User authentication and data management are supported by Supabase, offering secure and efficient management of user sessions and project data. Local Storage allows file storage and access, enabling users to save and load their code snippets with ease. To enhance the development experience, the addition of the Gemini API offers an AI-powered chatbot with intelligent code suggestions, debugging tips, and real-time coding tips. Together, these technologies make CCT a strong collaborative programming tool, applicable in learning and professional environments. Future additions, including scalability improvements, AI optimizations, and cloud-based file storage, will further improve its capabilities, making it a flexible and indispensable platform for developers and learners.

References

- [1] K. Jannes, B. Lagaisse, and W. Joosen, “Owebsync: Seamless synchronization of distributed web clients,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2338–2351, 2021.
- [2] F. Boronat, M. Montagud, P. Salvador-Llàcer, and J. Pastor, “Wersync: a web platform for synchronized social viewing enabling interaction and collaboration,” *Journal of Network and Computer Applications*, vol. 175, pp. 1–21, 2021.
- [3] M. M. Draz, M. S. Farhan, S. N. Abdulkader, and M. G. Gafar, “Code smell detection using whale optimization algorithm,” *Computers, Materials Continua*, vol. 68, no. 2, pp. 1919–1935, 2021.
- [4] F. Thabit, S. Alhomdy, A. Al-Ahdal, and S. Jagtap, “A new lightweight cryptographic algorithm for enhancing data security in cloud computing,” *International Journal of Intelligent Networks*, vol. 2, no. 1, pp. 18–33, 2021.

Appendix A: Presentation



Final Presentation

Collaborative Code Tool




Presented by:

Eldho George
Eswanth Sunil Dutt
Joe Joseph
Mohammed Kaif

Guided by:

Ms. Liya Joseph
Asst. Professor
Department of CSE

Problem Definition

- **Lack of Collaborative Platforms:** Traditional coding environments are often limited to individual workspaces, making it difficult for teams to work together seamlessly on the same project.
 - **Limited Support for Real-Time Compilation:** Many existing tools do not support real-time code compilation and feedback across multiple users.
 - **Inefficient Project Development Workflow:** Managing code files, directories, and project structures across multiple collaborators can be **challenging**.
 - **Reduced Interactivity and Engagement:** Current systems often fail to provide an engaging and interactive environment..
- 

Purpose & need

- **Limited Interactivity:-** Many educational tools do not offer real-time code execution, resulting in a passive learning experience.
 - **The absence of immediate feedback** such as real-time error detection and syntax highlighting can complicate the development process by making it harder for developers
 - **Collaboration Barriers:-** Current platforms often fail to support collaborative work effectively, preventing learners from benefiting from peer feedback and teamwork.
 - **User Role Management:-** A lack of defined user roles can lead to confusion in collaborative settings
-

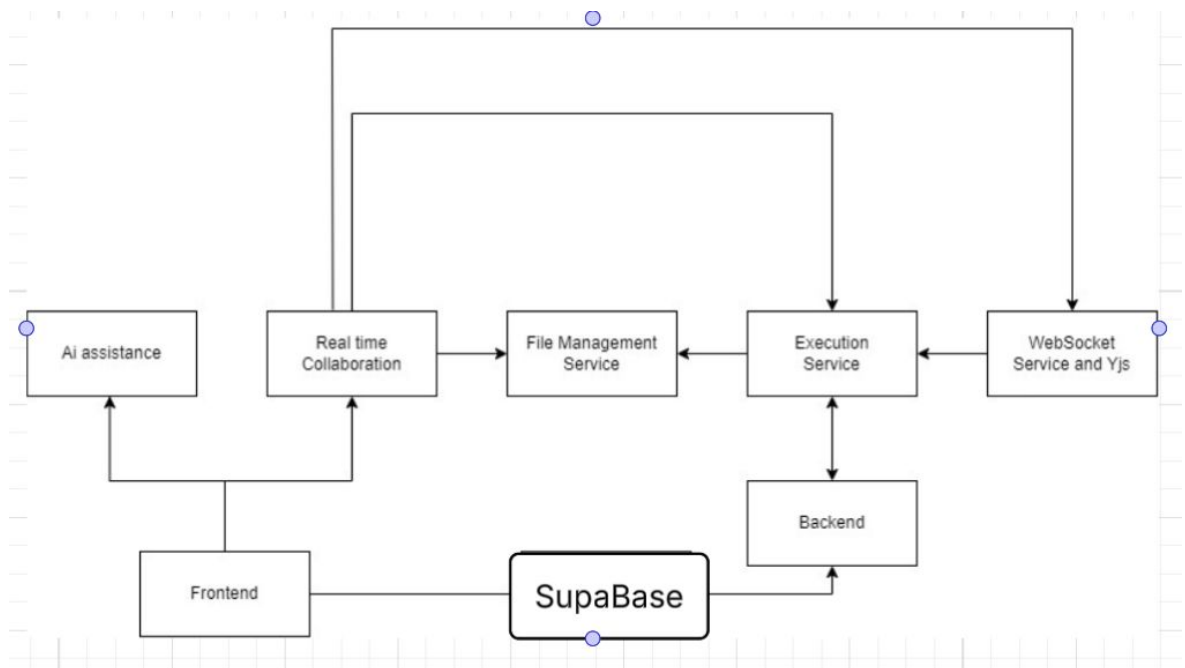
Project objective

- Create an engaging, web-based development and learning environment
 - Incorporate real-time code execution, error detection, and syntax highlighting.
 - Support various user roles to facilitate seamless collaboration among users
 - Provide a secure interface with user authentication for login and room access
-

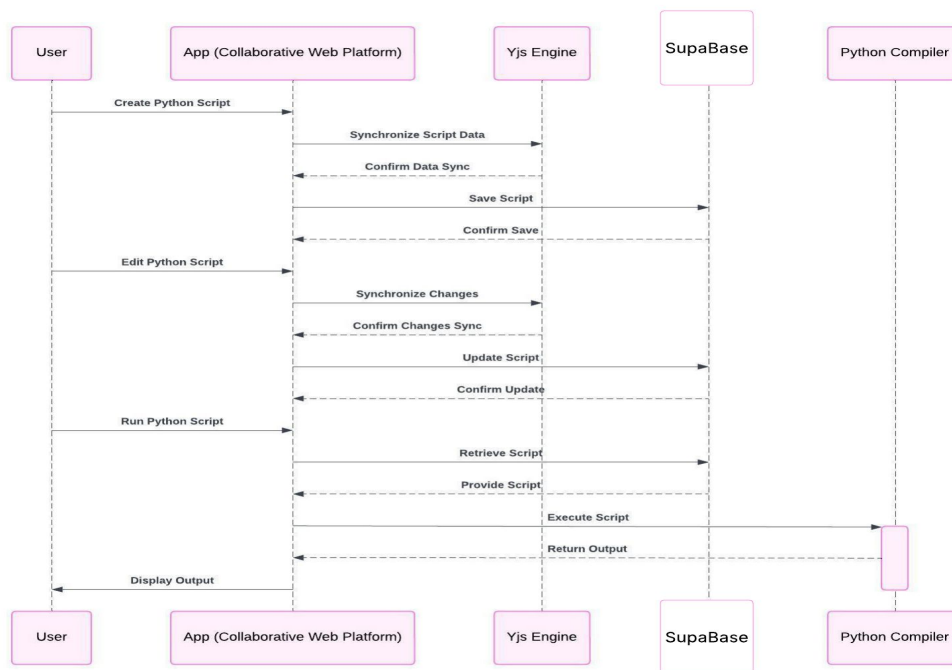
Proposed method

- **File Management:** Supports real-time collaborative editing and file management using **Yjs** for synchronization and **Cloud** for persistent storage.
- **Code Compiling:** Enables seamless execution of Python code in an isolated environment with real-time output and error feedback.
- **Concurrency:** Manages multiple users editing simultaneously without conflicts, using **WebSocket**-based real-time communication for synchronization.
- **Framework:** Built using **React.js** for the frontend, **Node.js with Express** for the backend, and **Monaco Editor** for the collaborative coding experience.
- **AI Assistance:** Acts as a virtual tutor, providing **language conversion**, **complexity reduction**, and **error detection** to enhance the coding and learning experience.

Architecture Diagram



Sequence diagram



File Module

Purpose: Represents a Python code file that can be collaboratively edited by multiple users in real-time using Local or Cloud storage.

Integration with Yjs: Enables live synchronization of file changes between users.

Key Features:

- Tracks the structure and contents of Python code files in real-time.
- Supports **file imports**, allowing users to reference and execute code from multiple files within a project.
- Coordinates with the **Yjs document** to synchronize file changes, ensuring all clients receive real-time updates.
- Works alongside a **compiler interface** for users to execute code and retrieve results after edits.

User Module

Purpose: Represents an individual user within the platform, managing authentication, authorization, and user-specific data.

Key Features:

- **Authentication & Authorization:** Ensures users are properly authenticated before joining a session.
- **Room Management:** Users are required to either create or enter an existing roomkey
- **Unique Room Id creation:** Make use of `uuidv4` function
- **Notifications:** Error messages, success confirmations, or alerts.

Execution Module

- **Purpose:** Manages the compilation and execution of Python code submitted by users during collaborative sessions.
- **Key Features:**
 - **Code Compilation & Execution:** Executes the Python code that users edit collaboratively.
 - **Execution Environment:** Manages a secure and isolated environment where the code is executed, ensuring no interference between users' sessions.
 - **Output and Error Capture:** Captures and displays execution results, including any runtime errors, directly to the users.

Collaboration Session Module

Purpose: Manages active collaborative editing sessions for Python files, facilitating real-time interactions between users.

Integration with Yjs: Utilizes Yjs for synchronizing changes and requires a **WebSocket** or **WebRTC** connection for efficient communication.

Key Features:

- Handles real-time **conflict resolution**, merging edits from multiple users without data loss.
- Manages **user presence**, showing who is editing and displaying cursors and selection highlights in real-time.
- Synchronizes updates through Yjs to ensure all participants see changes instantly and periodically saves the latest state to **storage**.

AI Assistance Module

- **Purpose:** Acts as a virtual tutor, helping users with code translation, complexity reduction, and error resolution, enhancing the learning experience.
- **API Integration:** The features are implemented with the help Gemini API which is connected to the interface
- **Key Features:**
 - **Language Conversion:** Translates programs from one programming language to another, aiding users in learning and writing code in multiple languages.
 - **Complexity Reduction:** Suggests optimized solutions or refactors the code to reduce complexity, making the code more efficient and easier to understand.
 - **Error Detection and Resolution:** Identifies syntax and logical errors in real-time and provides helpful suggestions to resolve them.

Assumptions

- **Users:** We assume that users are developers familiar and have prior knowledge of VSCode or similar IDEs.
 - **Internet Speed:** Assumes stable internet connection for seamless WebRTC and Socket.io-based real-time collaboration.
 - **Supported Browsers:** The web app will function optimally on modern browsers (Chrome, Firefox, Edge) with support for WebRTC and JavaScript features.
 - **Security:** Users will not share sensitive data in rooms, and basic encryption will be used for chat and collaboration.
 - **Project Scope:** The project is currently limited to file editing and real-time code collaboration for python only.
-

Risks & Challenges

- **Real-time Collaboration Lag:** High latency or low internet speeds may disrupt real-time collaboration.
 - **Concurrency Issues:** Multiple users editing the same file simultaneously might cause version conflicts.
 - **Security Concerns:** Protecting code files and chat messages from unauthorized access or leaks.
 - **Scaling:** Difficulty in scaling the tool to support large numbers of users or bigger codebases.
 - **Browser Compatibility:** Some features of WebRTC and Socket.io might not work uniformly across all browsers.
-

Expected Output

- **A functional web-based collaborative coding tool:** Users can join a room, work on files, and chat in real-time.
 - **VSCode-like Interface:** The users will experience a familiar code editor environment with multi-file and directory support.
 - **Real-time File Synchronization:** Changes made by one user are reflected in real-time to all other users in the same room.
 - **Chat and Code Suggestions:** A working chat feature alongside integrated AI for code suggestions.
 - **Scalable and Secure:** Basic user authentication in place for secure coding sessions.
-

Progress as of 50% evaluation

- **File management and interface**
 - **Code compiling (Execution Module)**
 - **Editor Interface**
 - **Real time editing from multiple users (in localhost).**
-

Progress as of 75% evaluation

- Real time editing from multiple users (collaboration).
 - Code conversion.
 - AI Assistance
 - User login with Supabase
-

Progress as of 100% evaluation

- Integration of all features into a single interface i.e the AI assistance feature was edited into the Editor interface
 - Implemented Chat room feature
 - Improvements to the Front end layout and appearance.
-

ScreenShots



Collaborative Python Editor

Username

Room ID

Generate

Create Room

Join Room

Files

main.py

```
1 print("this is inside main file!!!")
```

Output

▶ Run

this is inside main file!!!

Files

main.py

test_folder

file1.py

file2.py

```
1 import file1
2
3 file1.my_function()
4
```

Output

▶ Run

This function is from file1.



Conclusion

- The Collaborative Code Tool (CCT) offers an innovative approach to programming education by addressing key challenges such as interactivity, real-time feedback, and collaboration.
- By leveraging **Yjs** for real-time synchronization, **Cloud storage** for file management, and advanced **AI Assistance**, the platform provides a seamless, efficient, and interactive coding experience.
- With features like code execution, concurrency handling, and AI-powered tutoring, CCT is designed to enhance both individual and team-based learning experiences.
- This tool not only simplifies coding education but also fosters collaboration, making programming more accessible and engaging for learners at all levels.

References

- Gadban, F.; Kunkel, J. Analyzing the Performance of the S3 Object Storage API for HPC Workloads. *Appl. Sci.* **2021**, *11*, 8540.
- Nicolaescu, K. Jahns, M. Derntl, and R. Klamma(2015)
“Yjs: A framework for near real-time P2P shared editing on arbitrary data types,” in Engineering the Web in the Big Data Era. Cham, Switzerland: Springer, 2015, pp. 675–678
- I. F. Adams, N. Agrawal, and M. P. Mesnier, "Enabling Near-Data Processing in Distributed Object Storage Systems," in *Proceedings of the 5th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '13)*
- Boronat, F.; Montagud, M.; Salvador-Llàcer, P.; Pastor, J. (2021). Wersync: a web platform for synchronized social viewing enabling interaction and collaboration. Journal of Network and Computer Applications. 175:1-21. <https://doi.org/10.1016/j.jnca.2020.102939>

References

- Nicolaescu, K. Jahns, M. Derntl, and R. Klamma(2015)
“Yjs: A framework for near real-time P2P shared editing on arbitrary data types,” in Engineering the Web in the Big Data Era. Cham, Switzerland: Springer, 2015, pp. 675–678
 - Fursan Thabit,, Sharaf Alhomdy, Sudhir Jagtap(2021)“A new data security algorithm for the cloud computing based on genetics techniques and logical-mathematical functions” (International Journal of Intelligent Networks)
 - I. F. Adams, N. Agrawal, and M. P. Mesnier, "Enabling Near-Data Processing in Distributed Object Storage Systems," in *Proceedings of the 5th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '13)*
 - M. M. Draz, M. S. Farhan, S. N. Abdulkader, and M. G. Gafar, "Code Smell Detection Using Whale Optimization Algorithm,"(Computers, Materials & Continua)
-

Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes

Vision, Mission, Programme Outcomes and Course Outcomes

Vision: To become a Centre of Excellence in Computer Science and Engineering, moulding professionals catering to the research and professional needs of national and international organizations.

Mission: To inspire and nurture students, with up-to-date knowledge in Computer Science and Engineering, Ethics, Team Spirit, Leadership Abilities, Innovation and Creativity to come out with solutions meeting the societal needs.

Program Outcomes (POs)

The following are the Program Outcomes (POs) that engineering graduates are expected to achieve:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change..

Programme Specific Outcomes (PSO)

PSO1: Computer Science Specific Skills:The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.

PSO2: Programming and Software Development Skills:The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.

PSO3: Professional Skills:The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.

Course Outcomes (CO)

CO1: Model and solve real world problems by applying knowledge across domains.

CO2:Develop products, processes, or technologies for sustainable and socially relevant

applications.

CO3: Function effectively as an individual and as a leader in diverse teams and to comprehend and execute designated tasks.

CO4: Plan and execute tasks utilizing available resources within timelines, following ethical and professional norms.

CO5: Identify technology/research gaps and propose innovative/creative solutions.

CO6: Organize and communicate technical and scientific findings effectively in written and oral forms.

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO 1	2	2	2	1	2	2	2	1	1	1	1	2	3		
CO 2	2	2	2		1	3	3	1	1		1	1		2	
CO 3									3	2	2	1			3
CO 4					2			3	2	2	3	2			3
CO 5	2	3	3	1	2							1	3		
CO 6					2			2	2	3	1	1			3

3/2/1: high/medium/low

Figure 4.51: CO-PO and CO-PSO Mapping Table

Appendix C: CO-PO-PSO Mapping

Appendix C

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO 1	2	2	2	1	2	2	2	1	1	1	1	2	3		
CO 2	2	2	2		1	3	3	1	1		1	1		2	
CO 3									3	2	2	1			3
CO 4					2			3	2	2	3	2			3
CO 5	2	3	3	1	2							1	3		
CO 6					2			2	2	3	1	1			3

3/2/1: high/medium/low

Figure 6.1: CO-PO and CO-PSO Mapping