Grenze International Journal of Engineering and Technology, June Issue

CodeCollab: A Real-Time Collaborative Code Editor

Bansiddha Doge[1], S. A. Aitwade[2], S. S. Rokade[3], M. S. Dabade[4] and A. R. Surve[5]
[1]Student, Department of Computer Science and Engineering Walchand College of Engineering, Sangli-Miraj Road, Vishrambag, Maharashtra, India
Email: bansiddha.doge@walchandsangli.ac.in
[2-4]Asst. Professor, Department of Computer Science and Engineering Walchand College of Engineering, Sangli-Miraj Road, Vishrambag, Maharashtra, India
Email: swapnali.aitwade@walchandsangli.ac.in, sonlai.rokade@walchandsangli.ac.in, Manisha.dabade@walchandsangli.ac.in
[5]HoD, Department of Computer Science and Engineering Walchand College of Engineering, Sangli-Miraj Road, Vishrambag, Maharashtra, India
Email: anil.surve@walchandsangli.ac.in

***Abstract*— CodeCollab is a web-based real-time collaborative code editor designed to facilitate synchronous development among distributed teams. CodeCollab supports multiple programming languages and provides features such as multi-file editing, version control, file management and real time synchronization using Socket.io to enable a scalable and structured development workflow. The platform integrates checkpoint- based versioning, compiler support, file sharing and seamless communication through integrated chat to enhance remote teams productivity and coordination between them. With enhanced project management features, CodeCollab stands as a feasible solution for modern development environments that demand agility, reliability, and seamless collaboration.**

***Index Terms*— Real-time Collaboration, Web-based Code Editor, Socket.io, Version Control, File Management, Compiler Integration.**

I. INTRODUCTION

The transition towards remote and distributed software development teams has intensified the need for collaborative development environments. Traditional Integrated Development Environments (IDEs) are powerful in standalone environments but often lack real-time collaborative capabilities, which are crucial for modern development teams. While some aspects of collaboration have been addressed by cloud-based code editors such as CodePen, JSFiddle, and Replit, they do not offer formal project management or support for full-stack programming languages.

The application is supposed to resolve the problem by offering a web-based integrated development environment which features a wide range of functionalities. Unlike regular code editors, CodeCollab includes a tool for simultaneous editing, a folder structure system, multilingual compilation, and a version tracker. This combination of features gives developers the opportunity to work together during any stage of software development life cycle. Nowadays, software development depends largely on remote teams working in different locations. Tools helping to organize collaboration and cooperation effectively and simultaneously have become a must-have element.

Grenze ID: 01.GIJET.12.2.392_24

Despite numerous features provided by widely known editors, such as Visual Studio Code or Sublime Text, they fail to offer effective collaboration services and require additional applications, e.g., Live Share. CodeCollab offers everything required by teams, including an editor, compiler, chat, file manager, and version tracker. Thus, it helps developers communicate better, exchange information effortlessly, and work efficiently without switching contexts.

Some of the main objectives of CodeCollab include making it easier for teams to develop codes synchronously. There are several important features of the application:

CodeCollab supports editing code by several persons simultaneously. Participants are able to edit full-stack web application code. Changes will be reflected immediately.

CodeCollab provides a system for organizing files and folders. It will help users to work efficiently creating, renaming, deleting, and structuring files and folders in the current project.

CodeCollab uses the version control system, which means that users will be able to set checkpoints saving their code base state. It allows saving code stability and tracking its changes over time.

CodeCollab enables easy sharing and transferring files. Users will be able to upload files to the workspace and download project files. They will be able to share files with other people.

CodeCollab uses the WebSocket protocol, allowing users to interact with each other instantly.

In conclusion CodeCollab aims to provide an environment for teams to work together on code. CodeCollab helps teams work together efficiently and stay organized by providing real-time updates, file and folder management, version control and easy communication.

II. Literature Review

The work in reference [1] introduces CodeR, which is a real-time code editor that allows collaborative code editing. In this study, emphasis is put on the system's capabilities for allowing several users to collaborate on one piece of code and the problem of real-time synchronization, consistency, and managing latency.

The real-time code editor introduced in reference [2] offers several features like live code editing, file management, among others. This study focuses on the ability of the editor to help geographically separated teams become more productive through collaboration.

The study in reference [3] presents the development process of a real-time code editor that supports live editing, versioning, and synchronization. In this study, there is an emphasis on solving problems related to maintaining consistency and resolving conflict in real-time.

The study in reference [4] carries out a survey of some real-time collaborative code editors and looks at their strength and weakness, putting into consideration problems like merge conflict resolution, maintaining consistency, and the need for an intuitive interface.

Reference [5] examines cloud-based integrated development environments and gives information on their history, evolution, advantages, limitations, among other aspects. This work highlights benefits like scalability, accessibility, and real-time collaboration. Performance-related limitations are also mentioned in this study.

This study in reference [6] carries out a survey analysis of synchronization algorithms in real-time collaborative editing platforms. Problems like network latency and conflict during simultaneous edits are identified while proposing ways to improve future solutions.

Reference [7] provides an overview of AI-assisted IDEs, where it discusses the use of machine learning techniques in reducing the manual effort of coding. AI-based systems are discussed for their effectiveness in suggesting code completions and detecting bugs, improving the productivity of developers.

A guide for developing real-time collaboration applications using Websockets is discussed in reference [8]. Best practices for real-time data synchronization, among other things, are provided in this study.

Reference [9] examines role-based access control in collaborative platforms. How different types of users have roles assigned to them and what they are permitted to do is discussed in this study.

Reference [10] discusses the application of artificial intelligence technology in collaborative development tools. It is shown how AI assists in the code review process by identifying and suggesting bug fixes and improving developer efficiency.

III. Proposed Methodology

The overall architecture diagram of the real-time code editor is shown in Figure 1. One user can create a room ID and others can log in to the room using ID. Other users added into the session by the user who created room ID.

The code editor page is opened, where synchronous code editing can be performed and files can be uploaded, downloaded, saved and reverted to a version using the versioning system.
Following are the main features implemented in codecollab

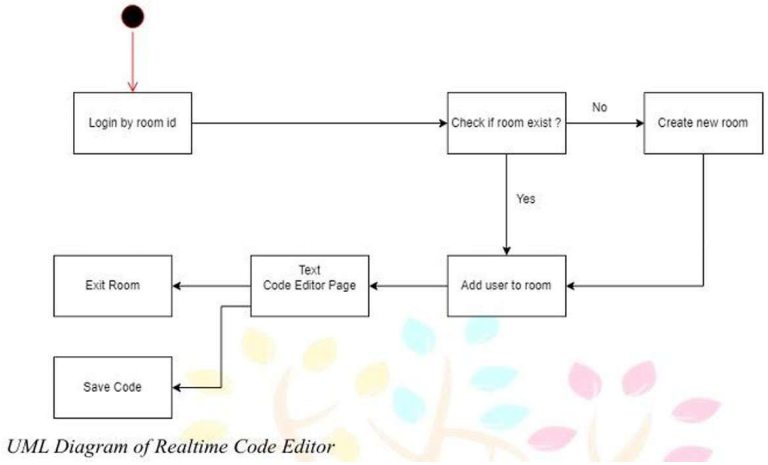

Figure. 1. UML Architecture Diagram of CodeCollab

A. Folder Structure & Multi-File Editing

CodeCollab is designed to provide the experience similar to any other desktop IDEs, such as Visual Studio Code. The user has access to multiple files and folders as well as real-time synchronization features. Several users can collaborate on separate files with instant sharing and viewing all the changes done to these files by others.
The application displays an expandable live file explorer that provides navigation through files and folders. One sees the current view based on the data kept in the database which is updated in a timely manner.
Editing, deleting, moving, renaming and creating files or folders becomes possible for all the team members collaborating on the project. All these actions are performed over a reliable Socket.io channel ensuring immediate updates of the file explorer displayed on all participants' devices.
MongoDB architecture provides a clear structure of all the files and folders. It includes the name, path, type and any hierarchy of these items. This structure makes it convenient to use files and folders without any problems in a long period of time.
The communication over Socket.io channel is provided in a very efficient way. Each user action concerning editing of file content or rearrangement of folders is transmitted to the server and forwarded to all the users collaborating on the current workspace.
However, real-time editing in case of multiple users in a single file causes some troubles and requires conflict detection and management. Therefore, the server performs tracking and operational transforms to detect potential conflicts and resolve them.
File tree on the front-end is refreshed after the changes occur in the database on the back-end side. In addition, the user is able to manipulate with files and folders, such as opening them and performing renaming operations within the user interface.

B. Checkpoint-Based Simple Versioning System

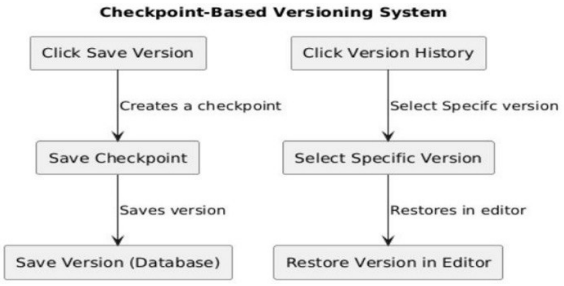

Figure.2. Checkpoint-Based Versioning Flow

To achieve stability in development and recover errors, CodeCollab uses version control as its basic function for capturing and managing snapshots of the codebase, which can act as recovery points, and can be referred to or rolled back whenever necessary.
Version Control with MongoDB All version information is stored in a special collection in the database of MongoDB. A checkpoint contains a copy of the current state of the entire project, including files and their structure. In this way, it is possible to save the correct project snapshot.
Creation of a Manual Version Whenever users decide to save a certain version of the project, they should use the interface button "Save Version." When this command is performed, the current state of the project is saved in the database. In addition, information about the time of the backup operation, the user who initiated it, and the identifier of the project will be saved in metadata.
Version History Viewing Users can view the entire version history in chronological order. Each entry in the history contains additional information, such as the time of creation and the user who created it.
Recovery to an Earlier State Users have the opportunity to revert to any version of the project. To do this, it is necessary to select the desired snapshot from the list of backups. Then, the current state of the project will be overwritten with the state from the selected snapshot.
Synchronization of Versions in Real Time To ensure the synchronicity between all the participants in the coding process, it is necessary to implement Socket.io broadcasting after each rollback operation. In this way, all collaborators will be able to work on a restored project version without any delay or confusion.

C. Support File Upload & Download

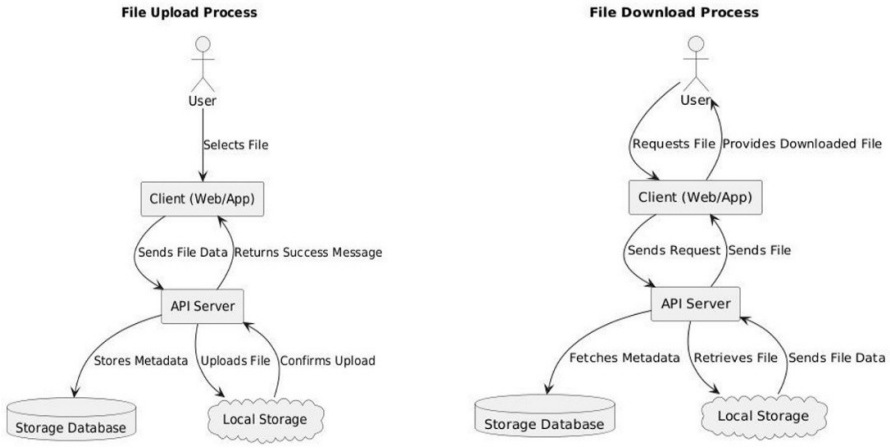

Figure 3: File Upload and Download Process

File Upload Process: The application comes with an advanced file upload functionality. This way, users can easily upload external files, assets, or code to the application. The process starts when the user chooses a file using the intuitive file input component. Once this happens, there will be a back-end processing request. For optimal file handling, Multer was used. It is a middleware for Node.js that helps with the parsing of multipart form data.
Depending on the configuration of the application server, the uploaded file will be saved either locally or in MongoDB's GridFS. Besides storing the actual file content, some important metadata will also be stored, such as file name, file size, upload time, and the user uploading the file. All users will be notified about a successful file upload, while all other collaborators will be updated through Socket.io. This way, all users in the project will have access to the newly uploaded file and can include it in the project file tree. File Download Process: Apart from file uploading, CodeCollab allows downloading of the files within the project. The download process begins when the user selects a file in the project's file tree. There will also be an option for downloading a particular file through a dedicated download button.
Once this button is clicked, a back-end file retrieving request will be performed. If successful, the user will be able to download the selected file to their computer. Optionally, Socket.io could notify the rest of the collaborators about the file being downloaded, increasing accountability. Real-Time File Sharing and Notifications Apart from providing the file upload and download functionality, CodeCollab allows sharing and notifications regarding changes within the file system.

Whenever a file is uploaded or downloaded to the project, it will be instantly reflected on the connected user's client app thanks to Socket.io. As soon as a new file is uploaded to the project, collaborators will immediately receive a notification about this event through in-app messages.

IV. IMPLEMENTATION

Node.js is the chosen framework to design the backend and enable scalable operation when handling requests from clients. As for the database for storing crucial information, MongoDB will be applied. It will include the data regarding projects, folders, user actions, and their version histories. Additionally, it will be necessary to use Socket.io to ensure real-time synchronization of changes and instant updates of the code made by users.

It will also be important to create a responsive interface to guarantee that users have access to all necessary functionality provided by CodeCollab. To achieve this aim, the front-end will be created with the use of React.js. It will provide the opportunity to develop components that can be added and interacted with dynamically. They will represent all the features available in the collaborative workspace, including code editing, file explorer, version history section, and notification system.

In order to save particular versions of projects, users will be offered to make checkpoint snapshots. The snapshots will be stored in a certain collection within MongoDB together with the relevant metadata. In case some version is required to be restored, all the changes related to it will be propagated to other active users through Socket.io. This feature will guarantee synchronization between participants of collaborative works.

Additionally, it will be necessary to provide several programming languages. The multi-language support will be implemented with the help of creating a special API layer. In this way, a separate compiler will be used for each language that will run in an isolated container. Containerization will guarantee the security of operations with code written in a particular programming language. The process of creating environments will be managed dynamically depending on the selected language.

This architecture guarantees robust operation, security, and scalability of CodeCollab. Now it is being realized in the process.

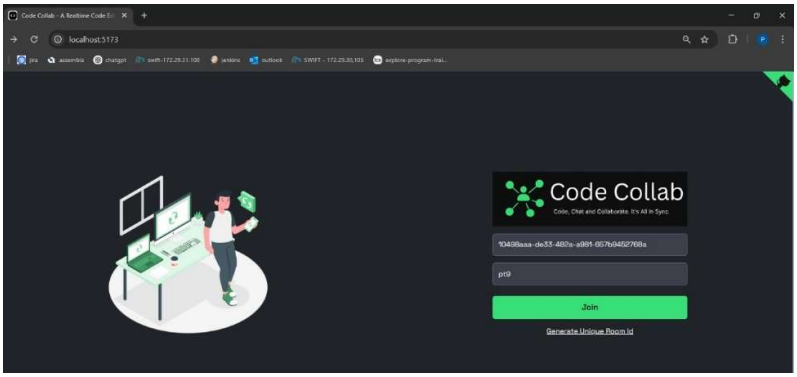

Figure 4: screenshot of Home page

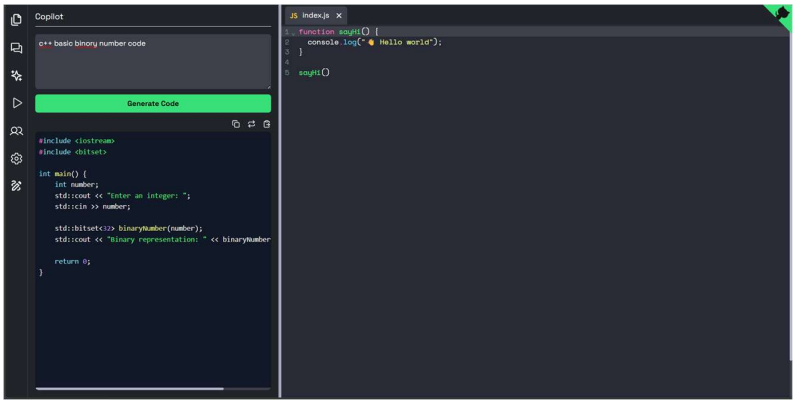

Figure 5: synchronous code editing

V. Evaluation and Results

This chapter analyzes the output results and performance of CodeCollab, an online tool allowing real-time collaborative coding for many programming languages. CodeCollab synchronizes all code modifications efficiently, supports more than 15 languages, organizes files and folders based on user activity, provides real-time chatting and supports downloading of code packages in zip form.

A. Real Time Code Synchronization

Synchronization Latency: In order to evaluate the performance of real time synchronization process in CodeCollab, we measured the time needed between making an edit and appearing it in other people's windows. With the help of Socket.io package and websockets in general, our software provides instant code changes without any noticeable delay in latency – average synchronization time is less than 100 ms in high-speed networks.

Conflict Management: In order to test conflict management, we simulated multiple users editing code at the same time. We used Operational Transformation algorithm for solving conflicts, and got the result showing that OT-based solution works well and doesn't allow conflicting edits.

B. Multi-language support Performance Evaluation

In this part we analyze the performance (accuracy, efficiency and latency) of compilation and execution of code samples in different languages:

Python: In case of server-side execution, Python is processed directly by the interpreter so the execution time is very short and close to zero.

PHP: The PHP code is executed by PHP-interpreter, which gives relatively quick results with minor latency.

Angular: In spite of not being a language, Angular compilation takes place pretty quickly for most of the components.

C. Files and Folders Management in Collaborative Code Editor

Evaluation of this process was conducted in terms of efficiency, convenience and speed of folder/file operations in collaborative environment. User may easily work with files and directories – create/delete files and folders, change names, group them into hierarchies. Folder/file creation/deletion/moving operations complete instantly within less than 120 ms. The system records all edits per file and displays active participants of collaboration and their actions, locking them out when needed. React-Tree component renders folders/files explorer in real time, helping user navigate in files and folders.

VI. Conclusion and Future Work

CodeCollab is a dynamic and collaborative web-based IDE that bridges the gap between single-user coding tools and modern team-based development environments. Designed with scalability, real-time interaction, and modularity in mind, it is well-suited for academic collaborations, hackathons, and remote software development projects.

In the upcoming development phases, several enhancements are planned to improve the platform’s usability and capabil- ities. One of the major updates includes integrating services like GitHub or GitLab, allowing users to directly import and manage repositories within CodeCollab. This would make it easier to synchronize codebases and maintain version control from inside the platform. Additionally, a role-based permission system is planned to ensure that users have specific access rights based on their roles, such as administrator, contributor, or viewer, thereby enhancing platform security and control.

To further support collaborative workflows, tools for com- paring and merging code versions will be introduced. These tools will help users visualize differences between file versions and resolve any conflicts more efficiently. Another significant improvement will be the addition of AI-assisted features such as code suggestions and error detection. These intelligent tools, powered by machine learning, will assist developers in writ- ing better code by offering recommendations and identifying issues in real time.

Together, these future updates aim to make CodeCollab a more powerful, user-friendly, and intelligent platform for collaborative coding.

References

[1] A. Kurniawan, C. Soesanto, and J. E. C. Wijaya, “Coder: Real-time code editor application for collaborative programming,” Procedia Computer Science, vol. 59, pp. 510–519, 2020.

[2] K. Aher and J. R. Mahajan, “Code Collab–Real Time Code Editor,” Adsul’s Technical Campus, Ahmednagar, 2023.
[3] A. Bhattacharjee and A. Das, “Design and Development of Real-time Code Editor for Collaborative Programming,” 2023.
[4] D. Sharma, A. Ghosh, and A. Kumar, “Collaborative Code Editing: A Survey,” Proceedings of the International Conference on Intelligent Computing and Communication, Springer, 2019, pp. 321-328.
[5] Z. Zhang and W. Qiu, “Cloud-Based IDEs: A Survey,” IEEE Access, vol. 8, pp. 112004-112021, 2020.
[6] M. Zhou and Z. Lu, “A Survey on Real-Time Collaborative Program- ming,” International Journal of Computer Applications, vol. 179, no. 34, pp. 40-45, 2019.
[7] H. Finkel and A. Berman, “AI-Assisted Code Completion and De- bugging: A Survey of Tools and Techniques,” Journal of Software Engineering and Applications, vol. 14, no. 3, pp. 99-110, 2021.
[8] D. Cohen and C. Daniels, “Building Real-Time Collaborative Platforms Using WebSockets,” Proceedings of the Web Development Conference, ACM, 2021, pp. 245-254.
[9] E. Santos and C. Alves, “Role-Based Access Control in Collaborative Software Development Platforms,” IEEE Software, vol. 39, no. 4, pp. 58-67, 2022.
[10] L. Pereira and A. Silva, “Leveraging AI for Code Review and Debug- ging,” Software Quality Journal, vol. 28, no. 2, pp. 401-417, 2020.