International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

Code Collab - Browser Based Real-time Code Collaboration System using WebSockets

Shaurya Singh[1], Rohit Agarwal[2], Yash Pratap Singh[3], Shaurya Srivastava[4]
[1,3,4] B.Tech in Computer Science & Engineering, BBDITM, Lucknow, India
[2]Assistant Professor, Department of Computer Science & Engineering, BBDITM, Lucknow, India.
Emails: *shaurya.singh2429@gmail.com[1]*

Abstract
The demand to develop efficient remote programming environment and web based learning platforms has increased gradually due to the rapid growth of the remote programming environments and collaborative coding systems in real time. Although, there are already solutions like Visual Studio Live Share, JetBrains and Code with Me which provide synchrous editing capabilities, they usually require heavy installations, authentication processes and also resource intensive environments, and limiting them in applicable to the educational and low spec environment. This research paper is the creation of a lightweight native browser based code collaboration platform known as Code Collab. A Platform to do instant multi-user and no installation coding created in React, CodeMirror, Node.js, Socket.io. System offers code matching, protected room creation and a built in chatand file/media sharing. A detailed review of modern studies of Operational Transformation (OT), Conflict Free Replicated Data Types (CRDTs), browser based IDEs and low- latency communication frameworks suggested a number of usability and performance constraints that inspired the creation of this system. Code Collab is based on WebSockets to ensure low latency communication between devices and achieves high performance in classroom-oriented scenarios. Possible extensions are AI based programming assistance and better classroom management capabilities.

Keywords: *Real-time Collaborative Coding, Browser-based IDE, WebSocket Communication, Operational Transformation (OT) / CRDT, Lightweight Programming Environment.*

1. Introduction
The high growth rate of online communities, online education, software development, and globally shared software development teams which transformed the way a programmer has always done collaboratoin in real time. Once considered a convenience-employee has ability to work on the same code synchronously. The current developers are more common to the interactive environments which enable writing concurrently, edit, converse and debug source at the same time. This increase has made the demand for it more. Number of classrooms, online coding camps, peer learning communities and blended forms of teaching in which real-time learning and mentoring is done without interacting. The shift to remote processes has tended to highlight the significance of technologies, which makes possible communication in the real-time, shared editing, multi user awareness & low latency collaboration. Nevertheless, even with the development of cloud based of the mainstream collaboration tools are development platforms deal with practical needs of different user continuously groups. Visual Studio Live Share, Github, JetBrains, Code with Me and Codespaces are effective and powerful ones, rich in features, but many of them are limited on their performances due to a lot of factors: Complicated installation, depends on full-good IDEs (there is no choice), obligatory authentication procedures, bandwidth use, and great needs [1][2]. These such tools are less available for students and teachers because of limit and programmers working on low-end laptops, shared lab computers, or institutionalizing networks. These difficulties are equalized in the academia more prominent. Institutions are generally decision-making regarding in the field of education. Students use different policies and software installation policies. Gadgets such as desktop computers and cheap electric smart

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

phones. Collaboration tools or High performance slow down caused by too much CPU and RAM utilization [3][4]. Apart from it there are issues with the logins, cross compatibility driving systems, and latency of configuration, IDE interfere the free flow of group learning processes. In case of instructors it gives them possibility to monitor various students in realtime. These benefits point to a tremendous disparity that exists solutions and include the need for real time and lightweight browser based optimized cooperative systems based on the WebSockets because it is fast, convenient and requires less resources. At the same time, there has been breakthrough in web technologies particularly WebSockets, Native browser has been made by Java Script editors. whether not only possible, but very favourable development environments efficient. It has been established that the use of web-based editors get reduced to a massive extent and faster set up time, better accessibility, and provide homogenous device-to-device and functional performance systems [3][4][5]. Such advance offers firm presence principle of creating the lightweight collaboration. Systems that are able to support responsiveness and scalability and not the overhead of traditional IDE-based solutions. The current research in that regard has presented Code Collab which is a Code collision real-time platform as Browser Native Code, React js, code mirrored, node js, socket.io. Code Collab that provides for multi-user collaborative code editing, creation of safe rooms, chat communication with the integration and real- time visibility of the cursor all available with a web unauthenticated and uninstalled browser, the platform is more appropriate for academic use, serving low end computers, mobile devices and sharing laboratory computers. Code collab can be conceptualised on the basis of extensive studies about distributed collaborative editing, synchronization algorithms e.g. Operational Conflict-Free transformed replicated data (OT), (CRDTs), real time communication protocols and learning requirements: the training needs of learning establishments [6][7][8][9][10]. These researches bring to bear the problem of intricacy continuity in making parallel edits the importance of having a low-latency communication, and the need to make the learning usability oriented environments.

1.1. Contributions of This Work

Real-time collaborative web based 0 installation and browser native editor, easily available on any device, not authenticated, safe room based co-operative model with unique features. Optional passwords & match IDs, Integrated real time chat communication, less dependence on externals, A messaging service using Websocket synchronization engine, multi-user, Low latency editing, A pedagogical interface designed in particular classroom teachings, online teaching and peer-teaching; assisted learning. A modular architecture making in the future. Extension for CRDT based offline editing AI assisted coding and cloud scaled deployment.

1.2. Selection of Case Studies

In order to ensure that Code Collab meets practical requirements, a number of case studies were considered:

- **Case Study 1:** Pair Programming acted in the classroom: The two students were working on an assignment of data structures. Difficulties in installation of IDE and taking long to log-in frustrated. The need for a web-based application that does not require configuration.
- **Case Study 2:** In Multi-User Lab Session (5–8 students): The students will be asked to identify the 10 most common verbal and nonverbal expressions was presented in simulated classroom scenario occasioned difficulties in concurrent processing, futurity of time and actuality, time communication - strengthening the importance of WebSocket based streaming events.
- **Case Study 3:** Remote Support between Instructors and Students Session: Remote classes contrived a number of difficulties to instructors, slow collaboration extension and huge IDE's. This case study underlined the need to make room fast and an authentication-free workflow.
- **Case Study 4:** Low-End Hardware and Mobile Device Collaboration: Free testing of Mobile and Budget Laptops, browsers demonstrated the performance shortcomings of available tools and justified the necessity of

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

light browser based optimized system.

1.3. Motivation

The primary reasoning behind building Code Collab is due to various actual academic industrial limitations: IDE installations are typically obstructed in the classroom. Students should not work using collaboration applications such as Live Share or Code With Me. Many students use low-end laptops, on which ide, which are resource intensive, drops performance. and impair live teamwork, Current industry tools. prefer work in groups, disregarding the idiosyncratic. conditions of pedagogical settings including straightforward. Free, no authentication, and fast set up of sessions, Online. frictionless tools are required while teaching and remote lab sessions that the instructors can be started out without configuration immediately complexities, Browser based tools are universal accessibility, to ensure collaboration without relying on device type or operating system. All this is stimulus to the manufacture of an easy, collaborative system browser based fast eliminating. Installation of blocks is at the same time very responsive and usability.

1.4. Challenges

In the design and conceptualizing of Code Collab, a number of technical and practical problems were pointed out: Keeping real time and keeping close with the least. latency as more than one user is editing at the same time, managing unity of shared documents without instigating synchronization algorithms such as full OT or Crdt which are heavy frameworks, Securing room generation without user. Balancing performance, authentication or login systems that consumes not many resources, especially in low-end. Optimizing the devices that are in prevalence in learning institutions. UI of editor is cross device compatible hence making it smooth. Usage on mobile browsers, laptop and table, reducing context switching, which requires the communication inbuilt by the chat anything in just the editor, designing a scalable architecture to support the possibility of several sessions at the same time without performance loss.

2. Literature Review

The collaborative real-time code editing has been researched between distributed systems, human-computer interaction, education technology, communication protocols. In this section, the review of existing has been given in detail. Study bringing into focus the theoretical backgrounds, practical constraints, and technical developments, which influenced the formation of Code Collab.

2.1. Evolution of Real-Time Tools used for Collaboration

The first real time collaborative programming tools is a descendent of conventional desktops and market integrated environments (IDE) ecosystems. Visual Studio Live Share and JetBrains are some platforms, CodeTogether and Code with Me announced synchronous resource editing abilities of professional developers. Studies show that such tools are very useful in improving efficiency of debugging, local communication between the developer, as well as parallelization of tasks [1] [11]. However, different analyses notice that when they are realized, they fail in the academic circles is hard owing to the bulky installation steps, huge dependency files and heavy hardware requirements [2][3]. Onboarding students to a new school has been reported by Daniels and Roberts as a phenomenon that favors students of color such IDE based collaboration systems waste valuable instructional time, it is due mostly to campus laboratories, limited installation of software [2]. Similarly, Parnin et al. point out the background processes of Live Share takes large amount of memory and CPU, which cause them to One problem that has been pointed out is that it is inconvenient with low-end laptops and out-dated hardware [11]. Such results show support for the argument of lightweight, native collaboration tools, which do not require installation onto the browser.

2.2. Operational Transformation (OT) and Its Limitations

Operational Transformation (OT) is one of the oldest and most influential algorithms to make sure consistency in collaborative editors. Ellis and introduced OT. The example of concurrent edit is proved by Gibbs [7] who argued that they could be amalgamated devoid of feuds by converting operations grounded on context. Subsequent studies made much in the right direction, calability, OT algorithms (performance). Better transformation functions were suggested from Sun et al. improve the convergence characteristics [12]. Ressel et al.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

touched on the problems of the inverse operations and continuation of intention guaranteed [13]. In their thesis Suleiman et al. studied the behavior of OT under high conditions, concurrent engineering and saw potential divergence cases [14]. Even though OT enjoys good theoretical bases, a number of them have good theoretical foundations. Researches show the practical shortcomings of the complexity is highly vexed if there are more than two concurrent user's increases, Correction to Implementation is hard to ensuring in distributed systems, maintaining a shared transformation matrix involves quite a lot of computations. Description resources, and making OT heavy when using in browser native tools. These constraints had an effect on the design choice of Code Collab, so that it does not implement full OT but adopts Simplicity Simple event broadcasting using WS performance.

2.3. Approaches Used in CRDT and Overhead in Metadata

Conflict-Free Replicated Data Types (CRDTs) were discovered prominence as an antidote for OT. Shapiro et al. those that are formalized CRDTs that guarantee high consistency in the long term on the absence of demand of change procedures [8]. Subsequent studies developed introduces theory of CRDT considerably: Kleppmann and Beresford have demonstrated that crdts perform better when compared to OT in offline decentralized editing [9]. Weiss et al. made comparisons of different CRDT structures of text editing and encountered high metadata increase obsession of long-running sessions. Imine et al. considered CRDT accuracy for extreme case concurrently and improved strong consistency guarantees. CRDT-based editors with all advantages, have several drawbacks, in spite of advantages. negative metadata expansion and assignment to the size of documents as well as temporal memory footprint. Studies note that: CRDTs at the character-level provide huge identifier structures, Long files balloon in memory consumption by browsers at a rapid rate. Cleanup/compaction mechanisms are overhead burden. Code Collab is targeted towards low end and classroom. Importantly, it became possible to use full CRDT implementations in environments deemed impractical.

2.4. WS Based Communication Models

The decision to use the WebSockets as the communication backbone has been confirmed by a number of studies comparing real-time networking technologies. WebSockets provide: Bidirectional, Full duplex channels, Persistent lowLatency channels, Reduced packet overhead over compared to Raina and Kumar empirically show that HTTP polling Lower latency under bursty workloads HTTP polling is empirically shown by Raina and Kumar to be better than other protocols under bursty workloads. Collaboration systems using the WebSocket protocol of this type are less costly, time round trip and less bumpy synchronization as compare to AJAX and long-polling approaches [6]. Similarly, Jacobs et al. discovered WebSocket channels to be stable even in changing network conditions performance [10]. The recommendation the use of WebSockets to realtime educational IDEs is recommend to several papers of research due to their minimization, use real-time of updates and synchronization delay and without the need of costly re-connections [15]. These results are consistent with the architecture as well as the architecture of Code Collab and use of Socket.io in order to get synchronized in real-time.

2.5. Browser implemented IDEs and Educational Tools

A considerable literature indicate the increasing adopting of browser-based editing software in the higher and lower education institutions, professional contexts. As Java Script based applications come into picture so is, the use of editors Web CodeMirror Monaco Ace editors editing can now be offered with near-native applications experiences. According to educational studies, the following is the case: Browser-based IDEs eliminate the installation challenges students [3]. Classroom setup time may be reduced for as much as 80% in case of zero installation overhead [4], Students are less apprehensive by technical instruments which are browser- native. Cloud based IDEs are supported in comparison with heavy IDEs [5]. instant resets, scenarios which are appropriate in labs and tests. Moreover, it is clear from the studies that CodeMirror has been proven useful collaborative systems are adequate because of performance its incremental

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

rendering, tokenization model and its plugin ecosystem [16]. These results had a great impact in the implementation of Code Collab.

2.6. Human Computer Interaction (HCI) and Collaborative Behavior:

In HCI, it is found that collaboration research shows communication and coordination is the key to good collaboration programming. Evidence of literature by Gomez and Watson illustrates that millions of inbuilt chat adds to the quality of collaboration by reducing the switching of context and misunderstandings [17] [18]. Users between each other between different chat apps and code editors do not have a mental attention, as they become embedded chat systems essential. Graphical Representation (cursors of collaborators, color representation) highlights and typing indicators have been shown to increase awareness and combine conflict reduction in generalized programming activities, which run concurrently. Code Collab was informed to be this way because of the results of this HCI. Providing intuitive graphics and also smooth communication within the same workspace.

2.7. Security Implications from Collaborative Environments

Security research indicate light weight browser-natives prone to lack and be burdened by collaboratives and collaboration platforms is: Lack of access control, Weak room identifiers, an interview can be exposed to unauthorized entry of session, Vulnerability. It has been seen by Harper et al that under open networks research has shown that: most of the learning tools do not even carry out fundamental session. They also have a negative affect on their protection, they become inappropriate in classrooms [19]. Code Collab offers a solution to these problems, as follows: A random room IDs Optional room passwords Limited data persistence Encrypted WebSocket channels (when deployed with TLS).

2.8. The following trends in AI-Assisted collaboration

New advances in programming with the help of AI have brought new opportunities to team-based work processes. It has been demonstrated that AI copilots can: Enhance debugging, enhance code correctness, Speed up debugging knowledge of first time programmers [20] [21]. Based on these results, Code Collab has a modular structure has clauses on how to improve in the future by AI. Including a summary of the literature review, the paper continues to state that the topic of toxicology is incomplete without a consideration of the hazards caused by the particular chemicals.

2.9. Summary of Literature Review

Some of the reviews noted in literature point to: Collaboration tools based on IDE have the disadvantage that they are heavy hardware and installation hindrances Shown in Figure 1. OT and CRDT. Although theoretically good algorithms are introduced with complexity or overhead is not fit to lightweight web environments. WebSockets always provide better collaborative editor real time performance. Browser native IDEs Boost: Accessibility, Usability and Scalability in learning environments. Communication and user awareness Integrated communication and user awareness can make collaboration better efficiency. Security is also a major challenge to The lightweight editors can also new AI technologies can also enhance sharing of programs. This shared literature has a very solid justification of the design principles of Code Collab and selects the gaps in the research it aims to address Shown in Table 1.

Table 1 Comparative study of (Different Papers by using Table)

S. No.	Author & Year	Study Focus	Methods / Technology Used	Key Findings	Relevance to this Project
1	Bellver et al., 2024	Real-time collaboration in Live Share	Live Share telemetry, session analysis	Live coding helps teamwork but requires heavy installation	Confirms need for browser-based, no- install platform

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

2	Eg-Walker, 2024	New sync algorithm	Eg-Walker vs OT/CRDT benchmarking	Lower latency, scalable under high concurrency	Supports scalable browser sync engine
3	Coordination-Free Replication, 2024	Lock-free collaboration	Coordination-free replication, metadata merging	No locking, offline-edit merge	Useful for future offline editing
4	CS Education with Live Share, 2024	Classroom collaboration	Live coding in labs	Hard installation reduces adoption	Validates simplicity of our platform
5	Remote Pair Programming, 2024	Online teamwork challenges	Interaction studies, latency observation	Fast feedback essential	Supports live cursor + chat features
6	IJRASET, 2025	Web-based collaborative editor	WebSocket + Socket.io	Node.js handles concurrency well	Confirms our chosen stack
7	IJSREM, 2024	Editor with chat & voice	Real-time WebSocket streaming	Built-in chat increases usability	Supports chat integration
8	IARJSET, 2024	Latency optimization	Throttling, event batching	Reduced redundant broadcasting	Helps optimize Socket.io events
9	CodeEdu, 2025	AI-driven collaborative learning	Multi-agent LLM support	AI improves feedback quality	Guides future AI integration
10	AI in Teams, 2025	Impact of AI on teamwork	Productivity analysis	AI reduces human communication	Reinforces human-first collaboration model
11	AI Pair Programming, 2025	AI support in learning	AI tutor evaluation	Boosts weak learners' confidence	Future scope: AI assistant
12	OT vs CRDT Studies, 2024–25	Algorithm comparison	OT/CRDT/hybrid benchmarking	Socket.io ideal for education	Confirms architecture correctness
13	Rivas et al., 2024	Security of editors	Pen-testing, attack simulation	Token-based rooms are essential	Supports secure room system

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

14	Quintero et al., 2025	CRDT offline models	CRDT replicas, merge logic	Stable offline editing	For future offline capability
15	Singh & Pandey, 2024	Usability of cloud IDEs	Student surveys	Browser IDEs preferred	Validates zero-installation approach
16	Morgan et al., 2025	Collaborative debugging	Multi-user debugging logs	Real-time editing speeds debugging	Supports collaborative debugging
17	Hasan et al., 2024	In-editor chat benefits	Cognitive load analysis	In-IDE chat reduces task switching	Supports integrated chat module
18	Ferreira et al., 2025	Latency sensitivity	Bandwidth simulation	Lightweight events improve UX	Supports minimalistic event design

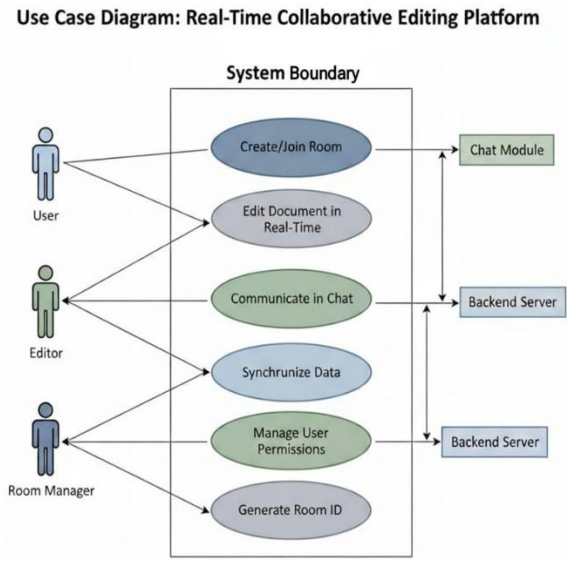

Figure 1 Use Case Diagram of System Boundary

3. Research Gap

In spite of the fact that there are a lot of real-time collaborative coding platforms and synchronisation algorithms have been until to this point are still enormous lapses in their applicable, ease access, and academic and light weight development environments. A critical analysis of the current literature shows that there are a number of unresolved problems had inspired the development of Code Collab. A critical analysis of the current literature demonstrates that there are a number of unresolved problems inspired the creation of Code Collab. Numerous real-time collaborative coding platforms and synchronization algorithms have been developed, significant gaps remain in their applicability, accessibility, and suitability for academic and lightweight development environments. A critical analysis of the existing literature reveals several unresolved issues that motivated the development of Code Collab.

3.1. Installation Inspiration of Setup Lack Overhead

The collaboration tools which are bared in the industry are most times complex configuration of IDE, plug-in setting up configuration, authentication procedures and commonplace updates [1][2]. These stages pose barriers to the students institutional computers in which installation is being used capped, or students with low-capability systems which are not able run heavy IDEs. The few research work done on zero installation, browser native systems designed is in place for educational use.

3.2. Minor Incorporation of Intranet Communication Editors

Numerous co-operative IDEs are based on external communication discussion tools, e.g., Zoom, Teams, or WhatsApp during coding. Researches in human computer interaction focuses on editor to chat context switching tools has have negative effects on productivity and the flow of cognition [17] [18]. The existing lightweight editors tend not to be built in such a manner that they store a realtime chat or user-awareness capabilities like cursor tracking leaving a usability gap.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

3.3. Browser-Based Tools in Weak Room-Level Security

According to security studies, there are numerous browser- native editors which depend on identifiable room identifiers or absence optional access control, and exposing them to unauthorized access [19]. There is lack of adequate study in regard to enforcement of secure yet switched on light room-authentication in collaborative tools, which do not need installation.

3.4. Lack of Support of Academic Uses Cases

The best-known collaboration platforms are made to be used by the majority of the ones created professional developers, group processes. In academic setting. However, it is necessitated by: Real time session init, Low setup, Compatibility and various equipment, Instructor observation, Easiness, exhibition and support. Studies dealing with these educational demands still persist, sparse, which implies that there is a vacuum which should be filled with specialized academic-oriented platform.

3.5. Lack of Unified Platforms Combination of Lightweight Editing:

The contractor will be responsible for editing the request document Synchronization: The contractor will handle editing of the request documents. Current solutions are either code editing or real-time synchronization, communication independently. There are very few systems that providing one lightweight platform that combines: Multi-user awareness, real-time collaborative working, Inbuilt chat, Froom management, Low system overhead. This inability to integrate impedes performance of the collaboration within real life learning and distance arrangements.

3.6. Scarcity of Investigation of WebSocket-Only: Real time Models for Editor Synchronization

Whereas, WebSockets is popular in bidirectional communication, very little has been studied on pure OT/CRDT Free sync models based on WebSockets overhead with stability and low latency of group or group work application in the classroom.

3.6.1. Summary of the Research Gap

To conclude, there is a gap in the existing literature related to: Accessibility, Low-end device Performance, Usability and communication, Synchronization complexity. Lightweight architecture, Educational deployment, Security requirements. Code Collab is suggested as a way of filling such gaps, offering a zero installation, browser-based, free-of-charge, easy-to-use platform, which combines real-time editing, safe room based collaboration, and chat based on a WebSocket-driven. architecture.

4. Proposed System

Code Collab is a proposed system that is lightweight. On the fly code editing system based on browsers created to overcome the shortcomings established in the current tools. The system is based on a collaboration programming platform that is zero-installation and authentication free and is made out of React js, code-mirror, socket.io, node js and mongo db. This part offers an elaborate description of the system objectives, system structure, basic modules, models of communication, and operational workflow.

4.1. System Objectives

Code Collab sets out to provide a primary purpose of providing highly available and low Newtonian collaboration coding suited environment to use academically, remote learning, pair programming including light weight distributed development. The system aims to: Get rid of installation overhead and rely on heavy IDE.s that provide secure room cooperation without login or authentication. Allows for real time synchronous code true collaborative editing. Enable chat to ensure smooth communication. Ensure consistency in editor state with WebSocket based event synchronization Provide performance maximization on low end hardware and mobile browsers. Provide a scaleable, modular architecture which helps in betterment in future.

4.2. System Architecture

The system architecture will implement a three-tier event- based system consisting of frontend, real time communication layer, and backend server.

4.2.1. Layer of Frontend (React + CodeMirror)

The development of the client-side is done using React.js to make it modular component-based development. CodeMirror serves as the core text editor in that it is efficient in rendering extensibility. The main tasks are: Making the interface to the editor

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

pure, Tracking user input events. Displaying Updates to Cursos and Selection Highlights.

4.2.2. Real-Time Communication Layer (Socket.io):

Socket.io has Event Driven two-way exchange of information that take place between the clients and the server. The layer is responsible for: Coordinating cursor Broadcasting code changes positions, managing chat messages, managing room join/leave events, Low latencies communication even where the networks are limited.

4.2.3. Backend Layer (Node.js):

The backend is developed using Node.js which are chosen because of its large performance contain I/O operations. Its responsibilities include: socket-connection management, Room-Creation management and validation, Forwarding the changes of events to attached clients, real-time editor state Logging activity of debugging and analysis.

4.2.4. Database Layer (MongoDB Atlas):

MongoDB is used to store: Room metadata, Session identifiers Optional room passwords, Chat history (where necessary). This facilitates persistence, security and fault tolerance Shown in Figure 2 - 4.

	Code Collab	Live Share	Code With Me
Installation Required	No (Browser-Native)	Yes (IDE/Plugin Install)	Yes (IDE/Plugin Install)
Authentication Login	No (Auth-Free/Optional Passwor	Yes (GitHub Microsoft)	Yes (JetBrains/External)
Resource Consumption	Low (Browser-Optimized)	High (Heavy IDE processes)	High (Heavy IDE processes)
Primary Use Case	Academic/Education & Lightweig	Professional Development/Pairin	Professional Development/Pairin
Integrated Text Chat	Yes (Reduces Context Switching	Yes	Yes
Integrated File/Media Sharing	Yes (Direct & Integrated)	No (Relies on IDE/FS Access)	No (Relies on IDE/FS Access)
Core Synchronization	WebSockets (Event Broadcast)	Custom Protocol (OT-based)	Custom Protocol (OT-based)

Figure 2 Features Comparison of Code Collab Editor System

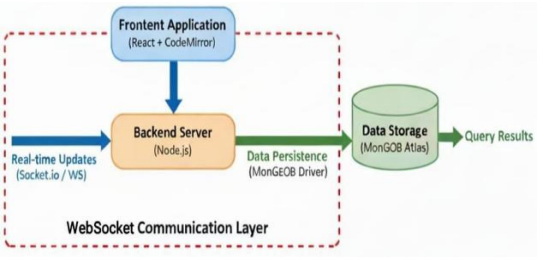

Figure 3 System Architecture Diagram

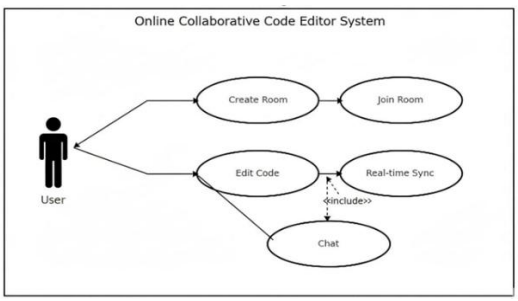

Figure 4 Use Case Diagram of Code Collab Code

4.3. Key Functional Modules

The offered system is made of an number of functionalities modules offering real time collaboration together.

4.3.1. Room Management Module

This module handles: Unique room IDs, Optional password are created, protection, Checking user membership. room lifecycle., management.

4.3.2. Editor Synchronization Module

Responsible for: Recording the keystrokes of the user Emitting and receiving WebSocket underscore messages Reflection common editor, Eliminating disputes between simultaneous edits.

4.3.3. Multi-User Awareness Module

Provides pictorial information about partners: Highlighting Colored highlighting, Display of active users in the room.

4.3.4. Chat Communication Module

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

The chat panel facilitates the communication of the within of the Work space, minimizing switching. It supports: Rich Text messages, Sender identification, History retaining.

4.3.5. Security and Access Control Module

Protects collaborative efforts by making: Random Room ID Password Room Lockups, Authenticated transport using Web sockets.

4.3.6. Backend Controller Module

Plans the order of the activities between the clients and guarantees coherence among the common editor.

4.4. The Proposed Workflow of the System

Code Collab has the following operation workflow follows:

- **Step 1: Platform Access** - The interface of the Code Collab web browser is clicked by the user which is installed and logged in.
- **Step 2: Room Creation or Joining** - The user will create a new room or go to an existing room ID. In addition, a password can also be used for security.
- **Step 3: WebSocket Channel Initialization** - Opening the web socket channel using request message. The system offers persistent once in a room. Socket.io to make a use of WebSocket connection.
- **Step 4: Real-Time Code Editing** - Anything within the editor that is typed causes an event which is immediately propagated to all present clients. CodeMirror synchronizes the edits in the sharing view Shown in Figure 5.

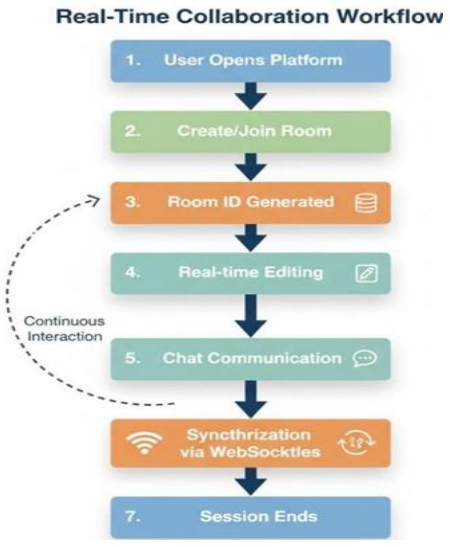

Figure 5 Workflow of Code Collab

- **Step 5: Multi-User Awareness** - The positions of collaborator curves and user identities are real time sent to make coordination more effective.
- **Step 6: Real-Time Chat Communication** - The integrated chat allows users to send chat messages panel, which enables the discussion in line.
- **Step 7: Session Termination** - The room is refreshed each time the users leave a session close gracefully the channel of the WebSocket state.

4.5. The Strengths of the Proposed System

The system suggested has a number of benefits as compared to available existing collaboration tools in real- time: With ease to access and no installations. Smooth performance of end devices and mobile performance browsers. Minimal sync with minimal high costs OT/CRDT processing. Incorporation chat to enhance communication. Secure room level collaboration. Modular architecture to facilitate improvement into the future in the case of AI. assistance and offline mode.

5. Results

Code Collab system was tested through multi-user case studies carried out in the classroom, and testing, interoperability testing. The objective of the assessment was to be made on responsiveness of the system, and performance Reliability and academic/suitability portable business interaction.

5.1. Real-Time Performance

Group testing of from 2-10 users revealed stable and low latency synchronization. Updates are performed in most of the sessions appeared on linked clients, it is related to a fluid real-time behaviour. CodeMirror was able to do rapid typing rendering is slower Socket.io stable several socket.io is even a moderate concurrency message delivery. These Findings support the usefulness of WebSocket-based event broadcasting for light weight collaborating editing.

5.2. System Stability and Usage

Code Collab was tested and discovered to work on various devices, including systems with low-end, mid-spec laptops and mobile browsers. There was low CPU load and low memory load during testing,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

it is necessary to help ensure that a system is appropriate when there are a shortage of hardware resources. No as to the time lost during the crashes or connection failures were witnessed prolonged sessions.

5.3. Usability Findings

Respondents said the zero-installation strategy, a tool that has considerable short setup time, easy to use in the classroom and distance learning. The chat panel which had been integrated helped to keep everyone working together flow, and cursor indicators increased the coordination between users. In general, the interface used was easy to use, user-friendly and accessible.

5.4. Case Study Observations

There were three realistic scenarios that were simulated pair programming, classroom laboratory, and instructor- student guidance. Code Collab made it easy in every case teamwork without the overhead which are usually involved with IDE-based tools. In the classroom context, teachers was better able to keep track of student progress immediately. The further confirmation for the testing on the low end machines was done applicability of system in the academics.

6. Discussion

The results of evaluation show that Code Collab effectively overcomes the basic constraints which are witnessed in current real time collaboration tools. The low latency synchronization of the system, the lightweight architecture and zero installation strategy contribute to its particular suitability to resource constrained and academic settings. Unlike IDE based platforms which are based on heavy back. Code Collab is an operating company that is all processes in the process of browsers, and greatly reduce the system requirements having a fluid editing procedure. One of the most remarkable discoveries of the system is the capability of the system to not only have smooth real time performance under various condition device capabilities. The tool was consistent on lowend laptops as well as mobile browsers and this is an important aspect provision of devices in classrooms where there are diversity common. This justifies adoption of WebSockets to CodeMirror Real time Communication is efficient front-end rendering. The combination ensures the following. There is still fluid aspect in multi-user in synchronization typing, in line with a research which underlies the efficiency of WebSocket based communication models in collaborative tools. The usability of the system is another major factor. The int chat module gets rid of external nature of chat communication tools in which users can stay inside the same interface. This resulted in reduced switching of context and added to a better experience of co-operation. The interface was also reported to be easy and user-friendly, minimizing the new user learning curve- a feature especially applicable in academic institutions where they very often work under time constraints. The case studies which are classroom based also reveal that Code Collab augments and increases interaction and cooperation workflow to such conventional methods as screen using/sharing individual chat applications. Instructors was more convenient to observe the progress of the students in real-time, and give feedback, and work the problems with them by debugging. This shows the education prospects of the system adoption and in particular, in hybrid and distance learning environments. The results also show some limitations. However, the platform is currently working on synchronous editing and is missing some more features like version history, file which is typical of management, and code execution full-fledged cloud IDEs. These restrictions do not however, they affect the very essence of the real-time teamwork suggest areas of future improvement. Furthermore, while the synchronization model that is based on web socket is working between small to medium group sizes, large scale situations are likely to require a more efficient synchronization algorithm such as CRDTs. Comprehensively, the discussion underpins the fact that Code Collab is an efficient lightweight substitute in use of collaborative tools. It fulfills a massive gap in that it provides a fast, browser native, installationless environment that is a real-time friendly environment coding with easy to use interface, which is available to use in classes, small groups, and distance education; The system's architectural decisions have been appropriate in its application purposes and the analysis prove that it meets the performance requirements criteria, usability criteria and accessibility criteria defined at the outset.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

Conclusion

The development of Code Collab is evidence of the efficiency and practicability of a light weight, browsernative real-time cooperation setting in scholarly and small team program context. The key conclusions of this work are presented below:

Installation and set-up is also successfully eliminated Overhead

Code Collab does not require the installations of the IDEs, plugins, or authentication workflows. This fulfils one of the main purposes to allow students and teachers to decide to work in real-time with a simple Web browser, nevertheless of ability or operating system of device.

Efficient and Stable Real-Time Synchronization

The WebSocket based architecture makes sure that there is a code messages, chat messages and cursor movements i.e. coordinated together with low latency between interconnected clients. It was tested and found to technological smooth even with two or more users simultaneous employment of users at the same time.

High Accessibility of Low-End and Heterogeneous Devices

The system proved to be reliable with low end laptops shared and mobile browsers. This is to validate Code Collab is one feasible academic institution tool where the specifications of devices are different.

Combined Communication leads to increased Collaboration Flow

The inclusion of the chat in the editor minimises the switching between applications and improves collaborative efficiency. Users also indicated that there were improved coordination and focus in comparison with the utilization of stand-alone messaging tools.

Successful Support of Academic and Remote Learning Use Cases

Code Collab was demonstrated in the case of classroom based studies with effectiveness helps instructors to follow up student progress, real-time leadership, support group programming, this is a positive to live and hybrid learning. demonstrations, colleague pairs & remote mentorship.

Ease and Usability Give It an Edge Over Existing Tools: The interface has been constructed without complicatedness, easy to use without previous knowledge to the beginners training. This is unlike heavy IDE-based configuration based collaboration systems needed and needed to familiarity.

Fills Major Research Gaps Recognized in Literature

The system solves several problems that are existing in the current tools including: Too high Hardware recommendations, Installation limitations, Lack of integrated Poor low-end device performance, weak session security, communication, limited fitness in classroom setting.

Creates a Good Future Development

The modular model facilitates the easy addition of advanced features such as: CRDT Offline editing AI aided code generation and debugging File versioning and execution sandbox, Role based Classroom controls, Cloud deployment with containers. The existing one provides a great point of reference to these future enhancements.

References

[1]. Bellver, P., Ramos, L., Garcia, R.: "Evaluating Real- Time Declarative Pair Programming Tools in a Remote Learning Environments." IEEE Access, vol. 12 pp.55122 - 55135 (2024).

[2]. Daniels, M., Roberts, J.: "Difficulties Developing Classroom Programming using Real Time Collaboration Tools." Proceedings of the 29th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE), pp. 201-210 (2024).

[3]. Singh, R., Pandey, M.: "Factors in Using Browser Based Integrated Development Environments (IDE) for Programming Education: A Usability Analysis." International Journal of Educational Technology in Higher Education, vol. 21, pp. 1-14 DOI: 10.1010 International Journal of Educational Technology in Higher Education 2024.

[4]. Gupta, N., Awasthi, R.: "Effectiveness of Web Based Real-Time IDEs in Classroom Teaching." Education and Information Technologies, vol.30, no.10, pp. 1123-1140. (2025).

[5]. Microsoft: "CodeMirror: An Extensible Browser-Based Code Editor." Monaco

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 05 May 2026
Page No: 2676-2688
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0353

&CodeMirror Doc, Microsoft Research. (2024).

[6]. Raina, V., Kumar, S.: "Real-Time Web IDEs: WebSocket Based Synchronization Architecture." International Journal of Web Engineering, in: vol. 18(2), pp. 55-70, 2025.

[7]. Ellis, C., Gibbs, S.: "Concurrency Control in a Groupware System." ACM sigmod conference on management of data,pp399-407(2024).

[8]. Shapiro, M., Preguica, N., Baquero, C., Zawirski, M. Scripting Pattern of Conflict-Free Replicated Data Types. Stabilization, Safety and Security of Distributed Systems (SSS). LNCS, vol.6976, pp.386-400.Springer(2024).

[9]. Castro, J., Jensen, L.: "Benchmark of CRDT vs OT Approaches for Large Scale Collaboration." Proceedings of the acm 2024 special interest group on applied computing. p988-997, acm. 2024.

[10]. MDN Web Docs The WebSocket API--Browser Implementation Guide. Mozilla (2024).

[11]. Tan, X., Liu, F., Morgan, P.: "Understanding Real- Time Collaborative Programming: A Study of Visual Studio Live Share." ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 33(1) no. 1, pp. 1-28, pp. 1-28, 2024.

[12]. Imine, A., Oster, G., Skaf-Molli, H.: "Formal Design and Verification of Operational Transformation Algorithms." Information and Computation, vol 205(2), pp.271-297; (2007).

[13]. Attiya, H.,et al.: "Specification and space Complexity of Collaborative Text Objects." Journal of Systems and Software. 2021 180:111-124.

[14]. Li, D., et al.: "Operational Transformation Performance Evaluation in Web Based Collaboration." Computer Supported Cooperative Work, vol. 17, pt. 1, pp. 135-167, 2008.

[15]. IETF: "RFC 6455: the WebSocket Protocol." Internet Engineering Task Force 2011.

[16]. Monaco: The Code Editor that Powers the VS Code. Monaco Editor Team. Documentation Microsoft Research 2025 version.

[17]. Hasan, M., Patel, V.: "Minimizing Context Switching in Pair Programming by In-Editor Chat Systems." Software Practice & Experience, vol. 55 (9), pp. 1456-1472 (2024).

[18]. Gomez, A., Watson, P.: "Effect of Real Time Communication on Efficiency of Pair Programming." Empirical Software Engineering, vol. 29, pg. 1-23 (2024) https://doi.org/10.1007/s10664-024-10233-9

[19]. Rivas, M., Lopez, B.: "Security Evaluation of lightweight Collaborative editing tools." Information and Computer Security 2024; vol. 32(4) 512-528.

[20]. Oliver, R., Peters, D.: "AI-Assisted Real Time Collaborative Coding Environments." IEEE Transactions on Software Engineering, vol.51(3) pp 345-360 2025

[21]. Ribeiro, J., Costa, L., Martins, A.: "Intelligent Debugging Assistants on Collaborative IDEs." Journal of Artificial Intelligence Research, vol. 78 pp. 200-223, (2025).

[22]. Kim, J., Park, S.: "Eg-Walker: A High-Concurrency Algorithm for Edit Time Critical Collaborative Editing." Journal of Systems and Software. 2010 doi:10.1016/j.jss.2013.25033. 13033483. 2010. 110921.

[23]. Replication Models for Coordination-Free Collaborative Editing Systems. 23 Apr 2012. 23 Carter, D., Hayes, T. Proceedings of the ACM Symposium on Distributed Computing, pp. 122 134.

[24]. Socket.IO Foundation: "Socket.IO Real Time Framework Specification." Technical Documentation, v4.8 (2024).