VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi - 590018, Karnataka

Project Work Phase 1 (BIS685)

on

"CodeNCollab: A Virtual Coding Environment"

Submitted in partial fulfilment for the award of degree of Bachelor of Engineering

INFORMATION SCIENCE AND ENGINEERING

Submitted by

NAGESH B C	**1BI22IS066**
NAMITH	**1BI22IS067**
PRAVEEN N PATIL	**1BI22IS083**
RAKESH S	**1BI22IS089**

Under the Guidance of

Prof. Priya N.V
Assistant Professor

BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V. V. Pura, Bengaluru – 560004
Department of Information Science & Engineering
2024-25

BANGALORE INSTITUTE OF TECHNOLOGY

V.V.Puram, K.R.Road, Bengaluru-560004

Department of Information Science & Engineering

CERTIFICATE

Certified that the Project Work entitled **“CodeNCollab:A Virtual Coding Environment”** carried out by **Nagesh B C (1BI22IS066), Namith (1BI22IS067), Praveen N Patil (1BI22IS083)** and **Rakesh S (1BI22IS089)** in partial fulfillment of the requirements for the VI Semester degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi during academic year 2025-2026. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Prof. Priya N.V	**Dr. Asha T**
Assistant Professor	Professor and HOD
Department of ISE, BIT, Bangalore	Department of ISE, BIT, Bangalore

DECLARATION

We, **Nagesh B C (1BI22IS066), Namith (1BI22IS067), Praveen N Patil (1BI22IS083)** and **Rakesh S (1BI22IS089)**, students of VII Semester B.E., in Information Science and Engineering, Bangalore Institute of Technology here by declare that the Project entitled **"CodeNCollab:A Virtual Coding Environment"** has been carried out by us and submitted in partial fulfilment of the requirements for the VI Semester degree of **Bachelor of Engineering** in **Information Science and Engineering** of Visvesvaraya Technological University, Belagavi during academic year 2024-25.

Date:

Place:Bangalore

ABSTRACT

In the rapidly evolving field of software development, **real-time collaboration** and integrated communication have become essential for improving productivity and code quality. Traditional development environments often fall short by lacking native support for **multi-user interaction** and requiring multiple external tools for communication, **synchronization**, and version control. This project presents **CodeNCollab**, a web-based, real-time collaborative coding platform equipped with an AI-powered assistant. The platform enables multiple users to write, edit, and execute code simultaneously within a shared environment, featuring real-time cursor tracking, synchronized views, and an integrated chat system. Developed using **React.js, Node.js, Express.js**, and **Socket.IO**, CodeNCollab includes a secure sandboxed code execution engine and leverages OpenAI APIs to provide intelligent code suggestions, syntax correction, and debugging support. Designed for developers, students, educators, and remote teams, **CodeNCollab** supports multiple programming languages, file import/export capabilities, and scalable deployment. By integrating collaborative tools and AI-driven assistance into a unified platform, this project aims to streamline the development workflow and foster an interactive, efficient coding experience.

CONTENTS

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
REST	Representational State Transfer
GraphQL	Query language for APIs
DB	Database
SQL	Structured Query Language
NoSQL	Non-relational databases (e.g., MongoDB)
ORM	Object-Relational Mapping
JWT	JSON Web Token (for authentication)
OAuth	Open Authorization (authentication protocol)
SDK	Software Development Kit
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
TS	TypeScript
SPA	Single Page Application
PWA	Progressive Web App
UI	User Interface
UX	User Experience
DOM	Document Object Model
RTC	Real-Time Communication
WebRTC	Web Real-Time Communication
SIP	Session Initiation Protocol (for signaling in VoIP)
RTP	Real-time Transport Protocol (media streaming)
SRTP	Secure RTP (encrypted media)
TURN	Traversal Using Relays around NAT
STUN	Session Traversal Utilities for NAT
ICE	Interactive Connectivity Establishment
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure
WS	WebSocket
WSS	Secure WebSocket

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, software development has increasingly shifted toward remote and distributed collaboration. Traditional code editors are designed for single-user workflows and lack real-time collaborative features. Developers often resort to using multiple disconnected tools for communication, file sharing, and version control. This fragmentation can hinder productivity and create barriers to effective teamwork. With the rise of online coding interviews, virtual classrooms, and global development teams, there is a growing need for unified platforms that support real-time coding and communication. Collaborative coding tools allow multiple users to code together in a shared environment. Integrated chat and video calling enhance communication and decision-making. Secure code execution is essential to protect users and systems. File import/export options further support seamless workflow management. This project aims to address these needs through an all-in-one platform—**CodeNCollab**.

1.2 Relation to Previous Work

Several studies, including “CodeR: Real-Time Code Editor Application for Collaborative Programming” and “Design and Development of Real-Time Code Editor”, have highlighted the need for efficient collaboration tools in programming environments. While platforms like VS Code Live Share and Replit exist, they often require installations, lack in-browser simplicity, or face latency issues. Our approach builds upon these existing systems but focuses on a purely browser based solution using Socket.IO, Node.js, and Monaco Editor, ensuring low-latency communication and ease of access without compromising on performance or features. Moreover, CodeNCollab integrates insights from real-time conflict resolution strategies mentioned in research such as CoRED and Miller’s collaborative editor, optimizing for seamless multi-user editing.

1.3 Importance

Enabling real-time code collaboration through a browser simplifies development and improves team synergy, especially in remote or distributed environments. CodeNCollab enhances the software development process by:

- Reducing time lost in context switching or screen sharing.

- Supporting educational use in coding workshops, hackathons, and live mentoring.
- Increasing accessibility by removing the need for heavy installations or setups.
- Supporting real-time messaging, code execution, and version saving within a single platform.

1.4 Present Developments

Currently, we are building CodeNCollab using a modular architecture where users can:

- Create and join private coding rooms.
- Collaboratively edit code with live synchronization using WebSockets.
- Chat in real time for communication and coordination.
- Execute code within the platform using backend execution services.
- Use syntax highlighting and intelligent suggestions for multiple languages.
- Ongoing enhancements include version control integration, role-based access (e.g., admin, viewer,
- editor), and persistent session saving. We're also integrating a basic authentication system using
- JWT to secure user access and streamline room management.

CHAPTER 2

Literature Survey

A literature survey or a literature review in a project report is that section which shows the various analyses and research made in the field of your interest and results already published, taking into account the various parameters of the project and the extent of the project.

2.1 INTRODUCTION

Real-time collaborative code editors have become a significant focus in software development due to the rise of distributed teams and remote work. Traditional development environments often fail to provide seamless collaboration, leading to communication delays and inefficient workflows. Several research efforts have aimed at improving real-time collaboration by integrating technologies like Operational Transformation (OT), Conflict-Free Replicated Data Types (CRDTs)

WebSocket-based synchronization to ensure low latency and high consistency. This literature survey focuses on five significant works related to collaborative code editing: CodeR, Real-Time Collaborative Code Editor Application, Design and Development of Real-Time Code Editor, Miller's Real-Time Collaborative Editor, and CoRED. Each paper proposes unique methodologies for improving user experience, synchronization accuracy, and ease of collaboration. By reviewing these contributions, CodeNCollab aims to implement an efficient, browser-based collaborative code editor with live synchronization and execution support.

2.2 SUMMARY OF PAPERS

2.2.1 CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural LanguageProgramming

Authors: Feng, L., Yen, R., You, Y., Fan, M., Zhao, J., and Lu, Z.

Published Year: 2023

Methodology:

This paper introduces CoPrompt, a collaborative natural language programming platform that enables prompt sharing and referencing among users. It supports real-time collaboration by allowing multiple participants to create, edit, and reuse natural language prompts for AI programming tasks. The system is designed to improve teamwork efficiency by maintaining a shared prompt repository and providing tools to track prompt provenance and usage context.

Drawbacks:

CoPrompt's reliance on natural language prompts can introduce ambiguity in programming tasks, potentially leading to errors. The platform currently focuses primarily on prompt sharing without integrating deeper debugging or version control features. Scalability in handling large teams or complex projects is not extensively discussed.

2.2.2 MeetScript: Designing Transcript-Based Interactions to Support Active Participation in Group Video Meetings

Authors: Chen, X., Li, S., Liu, S., Fowler, R., and Wang, X.

Published Year: 2023

Methodology:

MeetScript is a system designed to enhance active participation in group video meetings through transcript-based interaction. The system automatically generates meeting transcripts and allows participants to interact directly with the text, adding comments, highlights, and action items. This transcript-centric approach aims to reduce the cognitive load during meetings and promote inclusive.

Drawbacks:

MeetScript's effectiveness depends on accurate speech-to-text transcription, which can be error-prone in noisy environments or with diverse accents. It primarily supports post-meeting interaction and may not fully support synchronous collaboration or real-time decision-making workflows.

2.2.3 Self-Collaboration Code Generation via ChatGPT

Authors: Dong, Y., Jiang, X., Jin, Z., and Li, G.

Published Year: 2023

Methodology:

This work explores a novel approach where ChatGPT engages in self-collaboration to generate code iteratively. The model alternates between different roles (e.g., coder and reviewer) within a single session to improve code quality through self-refinement and error correction. The approach leverages AI's ability to simulate collaborative coding internally to produce more reliable software outputs.

Drawbacks:

The self-collaboration model relies heavily on the AI's internal logic without external validation, which can propagate errors if not properly guided. It may struggle with complex software projects requiring domain-specific knowledge or extensive testing.

2.2.4 A Collaborative Code Platform with Advanced AI Features and Real-Time CollaborationTools

Authors: Gaikwad, A., Agrawal, M., Goyal, J., Goyal, M., and Vinod, D. F.

Published Year: 2024

Methodology

This paper presents a web-based collaborative coding platform integrating advanced AI capabilities such as code completion, error detection, and refactoring suggestions. It supports real-time multi-user collaboration with live chat, version control, and an intuitive interface designed for seamless teamwork. The system is built to enhance productivity by combining human and AI efforts.

Drawbacks

The platform's AI features may require significant computational resources, impacting performance on low-end devices. Additionally, security and privacy concerns related to cloud-based code storage are not deeply addressed.

2.2.5 A First Look at Developers' Live Chat on Gitter

Authors: Shi, L., Chen, X., Yang, Y., Jiang, H., Jiang, Z., Niu, N., and Wang, Q.

Published Year: 2021

Methodology

This study analyzes live chat interactions on Gitter, a developer communication platform, to understand collaboration patterns, social dynamics, and information flow among software developers. Using data mining and qualitative analysis, the paper identifies common discussion topics and the role of chat in resolving coding issues collaboratively.

Drawbacks

The study is observational and limited to Gitter's user base, which may not generalize to other collaboration platforms. It does not propose system improvements or interventions based on findings.

2.2.6 CODEGEN: A React-Based On line Coding Workspace with Advanced Features and Execution.

Authors: Fathima, G., Kaif, M., and Nadeem, M.
Published Year: 2023

Methodology

CODEGEN is an online coding environment built with React that provides users with features such as syntax highlighting, real-time code execution, and debugging tools. It supports multiple programming languages and emphasizes real-time collaboration with shared coding sessions and chat. The workspace is designed for ease of use in educational and professional settings.

Drawbacks

CODEGEN currently lacks extensive integration with version control systems and has limited support for large-scale project management. Real-time collaboration may face latency issues under heavy network load.

2.2.7 Real-Time Collaborative Coding in a Web IDE

Authors: Verma, R., Nalisnick, E., and Naesseth, C. A.

Published Year: 2023

Methodology

This paper proposes a web-based IDE that supports real-time collaborative coding using operational transformation (OT) techniques to maintain consistency across multiple users. The platform features live code editing, syntax checking, and integrated communication tools, aiming to provide a seamless shared coding experience for distributed teams.

Drawbacks

The system's reliance on OT can lead to complexity in conflict resolution for deeply nested code edits. It may not support advanced IDE features like refactoring or deep semantic analysis.

2.2.8 Collaborative Virtual Reality Application for Interior Design

Authors: Kaur, M., Azrour, M., and Sinha, K.

Published Year: 2023

Methodology

This paper presents a collaborative virtual reality (VR) application aimed at interior design teams. Users can interact in a shared 3D space to modify room layouts, furniture placement, and decoration in real-time. The platform supports voice communication and gesture controls to facilitate natural collaboration in a virtual environment.

Drawbacks

The VR application requires high-end hardware, limiting accessibility. There are also challenges related to user interface intuitiveness and potential motion sickness during extended use.

2.2.9 On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies

Authors: Wong, A. Y., Chekole, E. G., Ochoa, M., and Zhou, J.

Published Year: 2023

Methodology

This paper examines container security through threat modeling and attack analysis, proposing a set of mitigation strategies for containerized environments. It evaluates common vulnerabilities and attack vectors, providing best practices to enhance container isolation, authentication, and runtime.

Drawbacks

While comprehensive, the paper focuses primarily on threat assessment without offering new security tools or automated defenses. Implementation details of proposed mitigations are high-level, requiring further practical validation.

2.2.10 From Code to Container: Understanding the Importance of Container Images

Authors: Sinha, K., Swaminathan, K., Tambe, T., Zhang, J. J., and Buyuktosunoglu, A.

Published Year: 2024

Methodology

This paper investigates the critical role of container images in the software deployment lifecycle, focusing on how container images bridge the gap between code and runtime environments. It analyzes container image creation, layering, and management practices to highlight their impact on security, performance, and reproducibility. The study combines empirical evaluation of popular container image tools with a threat analysis to understand vulnerabilities inherent in containerization.

Drawbacks

The research primarily focuses on container image creation and security at a conceptual and architectural level, lacking extensive experimental validation across diverse real-world applications. It also does not deeply explore container orchestration or runtime security beyond image concerns.

CHAPTER 3

Problem Statement

In modern software development, teams often struggle with **fragmented workflows** caused by switching between multiple tools for **code writing, execution, communication, and collaboration**. This leads to **reduced productivity**, **miscommunication**, **version control issues**, and **delays in development**. There is a lack of a **unified platform** that enables developers to **collaboratively write, execute, and review code in real-time** while also allowing **seamless communication** through built-in **chat and video features**.

CHAPTER 4

Project Planning

This project aims to develop CodeNCollab, a secure, web-based real-time collaborative coding platform with integrated communication and AI assistance. Below is the detailed project planning, including objectives, scope, phases, team structure, tech stack, and risk handling.

1. Project Objectives

- Build a web application where multiple users can collaborate on code in real time.
- Integrate features like chat, video calling, file import/export, and AI-based code suggestions and debugging.

2. Scope of the Project

In Scope: Real-time multi-user editing, sandboxed code execution, live chat, video conferencing, AI assistant, multi-language support, and file management.

Out of Scope: Native mobile application and integration with external desktop IDEs.

- Requirement Analysis – Understanding user needs and planning features
- UI/UX Design – Designing intuitive and responsive user interfaces.
- Frontend Development – Building interactive client-side components.
- Backend Development – Setting up server logic, databases, and API endpoints.
- Real-Time integration – Implementing live code sharing, chat, and video features.
- AI Integration – Adding intelligent coding support via OpenAI APIs.
- Testing & Optimization – Ensuring reliability, performance, and security
- Deployment & Documentation – Hosting the app and preparing user guides.

3. Technology Stack (With Explanations)

Frontend

- React.js – To build fast, modular, and reusable UI components.
- Tailwind CSS – For efficient and responsive styling with utility classes
- Redux / Context API – For managing global state such as user sessions and shared documents.
- CodeMirror / Monaco Editor – Embeds a powerful in-browser code editor with syntax highlighting and language support.

Backend

- Node.js – Executes JavaScript on the server for scalable backend logic
- Express.js – Handles API requests, middleware, routing, and server-side functionality
- JWT (JSON Web Tokens) – For secure user authentication and authorization.

Real-Time Features

- Socket.IO – Enables low-latency communication for real-time editing and chat.
- WebRTC – Allows peer-to-peer video and audio calls directly from browsers.

AI & Code Execution

- OpenAI API – Provides intelligent code suggestions, auto-completion, and error fixing.
- Docker – Creates isolated containers to securely execute user code in multiple languages.
- Judge0 API (optional) – An open-source code execution API to support multiple languages in a sandboxed environment.

Database & Storage

- MongoDB – NoSQL database to store user data, session history, and project files.
- Cloudinary / Firebase Storage – For managing file uploads like PDFs, images, or exported code files.

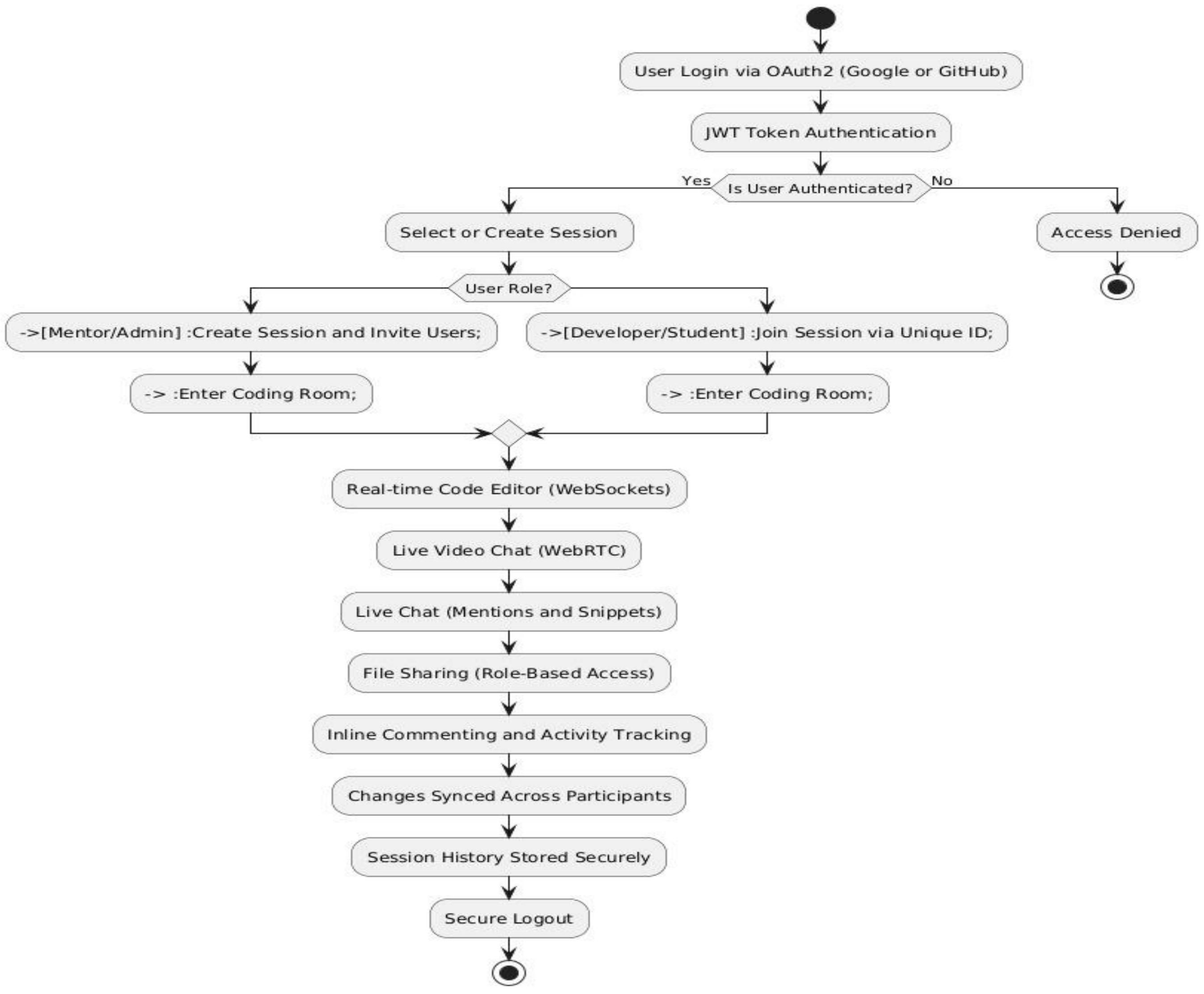
User Login via OAuth2 (Google or GitHub)
JWT Token Authentication
Is User Authenticated?
Yes
No
Select or Create Session
Access Denied
User Role?
->[Mentor/Admin] :Create Session and Invite Users;
->[Developer/Student] :Join Session via Unique ID;
-> :Enter Coding Room;
-> :Enter Coding Room;
Real-time Code Editor (WebSockets)
Live Video Chat (WebRTC)
Live Chat (Mentions and Snippets)
File Sharing (Role-Based Access)
Inline Commenting and Activity Tracking
Changes Synced Across Participants
Session History Stored Securely
Secure Logout

Figure 1:Technology Stack

- **Authentication**
- Firebase Auth / Passport.js – Handles secure user login and session management.
- OAuth 2.0 – Enables third-party login (Google, GitHub) for convenience.
- Hosting & Deployment
- Vercel – Hosts and serves the frontend with fast global delivery.
- Render / Railway / AWS EC2 – Used for backend and database hosting.
- NGINX / PM2 – Ensures stable deployment, load balancing, and process management.

Version Control & CI/CD

- Git & GitHub – For source code management and version control.

- GitHub Actions – Automates build, test, and deployment pipelines.
- **4. Team Roles and Responsibilities**
- Project Manager – Coordinates tasks, deadlines, and progress tracking.
- Frontend Developer – Designs and implements user interfaces and editor components.
- Backend Developer – Develops APIs, server logic, authentication, and database management.
- AI/ML Engineer – Handles OpenAI integration and smart assistant functionalities.
- Full-Stack Developer – Assists in connecting frontend and backend systems.
- Tester/QA – Conducts functional, security, and performance testing.

5. Risk Management

- Integration Issues – Delay or bugs during AI, video, or real-time tool integration.
- Security Risks – Threats from untrusted code execution or unauthorized access.
- Scalability Problems – Performance bottlenecks with multiple users.
- Mitigation – Use containerized execution (Docker), secure auth (JWT), regular testing, and backup strategies.

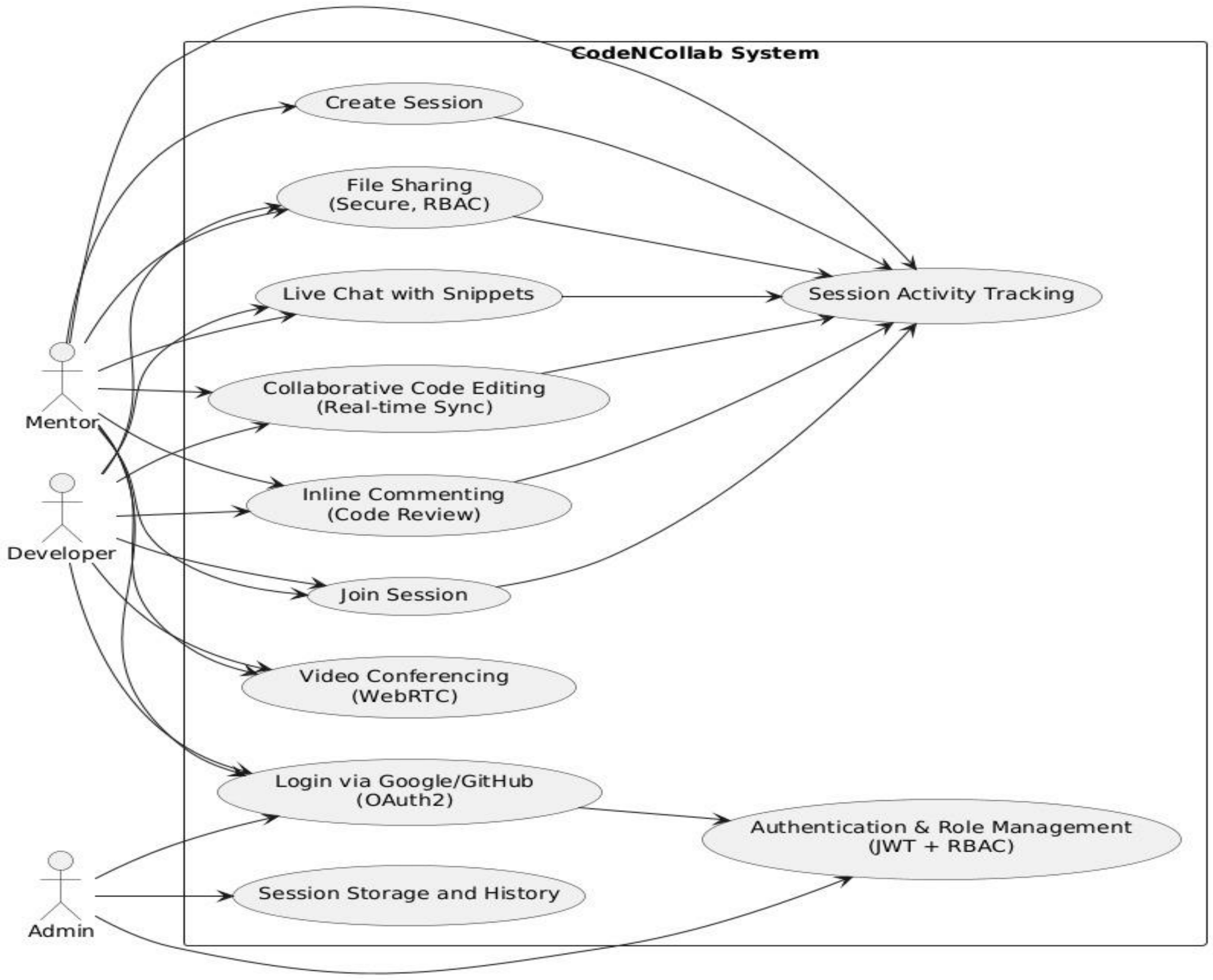
CodeNCollab System
Create Session
File Sharing (Secure, RBAC)
Live Chat with Snippets
Collaborative Code Editing (Real-time Sync)
Inline Commenting (Code Review)
Join Session
Video Conferencing (WebRTC)
Login via Google/GitHub (OAuth2)
Session Storage and History
Session Activity Tracking
Authentication & Role Management (JWT + RBAC)
Mentor
Developer
Admin

Figure 2:Use Case Diagram

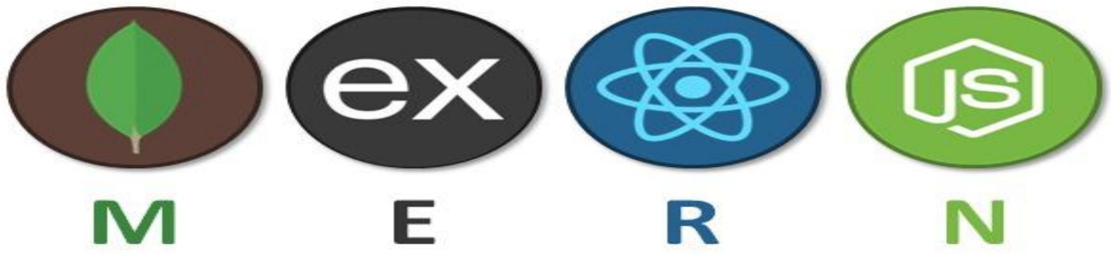

Figure 3:Tech Stack Flow

CHAPTER 5

System Design

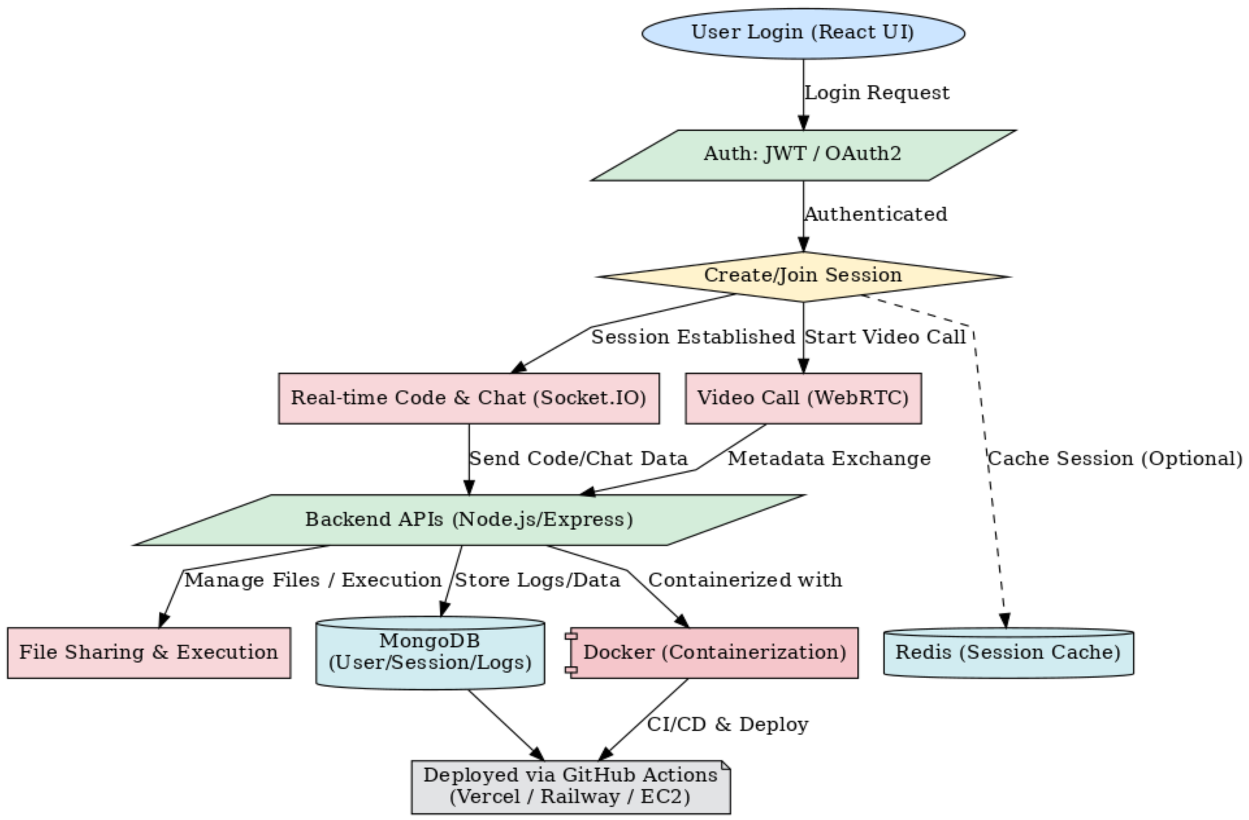

Figure 4:System Design

The system design of CodeNCollab outlines how different components interact to enable secure, real-time collaborative coding with integrated chat, video calling, and code execution.

1. User Login (React UI)

- The user accesses the frontend interface built with React.js.
- They initiate a login request using email/password or a third-party provider.

2. Authentication (JWT / OAuth2)

- The server validates usefr credentials using **JWT** (for session-based auth) or **OAuth2** (for Google/GitHub login).
- Once authenticated, users are allowed to proceed.

3. Session Management

- Users can **create or join a coding session**, which serves as a shared environment.
- The session is securely initialized and may optionally be **cached using Redis** to speed up performance and recovery.

4. Real-Time Code and Chat (Socket.IO)

- Socket.IO enables low-latency real-time updates for collaborative code editing and instant chat messaging.
- Data is synchronized across all connected users in the session.

5. Video Call (WebRTC)

- WebRTC handles **peer-to-peer video and audio calls**, allowing users to communicate visually while coding.
- Metadata (e.g., session IDs, user streams) is exchanged to initiate and maintain the call.

6. Backend APIs (Node.js / Express)

- The backend processes requests for file handling, session updates, user actions, and AI integration.
- Express routes manage RESTful communication between frontend and services.

7. File Sharing and Execution

- Users can **upload, share, and execute files** securely.
- Code execution may be handled in isolated environments for safety.

8. MongoDB (User/Session/Logs)

- MongoDB stores persistent data such as user profiles, session history, and chat/code logs.
- Offers flexibility through its JSON-like document model.

9. Docker (Containerization)

- Code execution is handled inside **Docker containers** to isolate user code and prevent server risks.
- Supports multiple languages and safe sandboxing.

10. Redis (Session Cache - Optional)

- Used to cache active session data temporarily for faster access and smoother reconnections.
- Ideal for scaling the platform under high user load.

11. Deployment (GitHub Actions + Vercel/Railway/EC2)

- CI/CD is managed using **GitHub Actions** to automate testing and deployment.
- The frontend is deployed via **Vercel**, while the backend and Docker services are hosted on **Render, Railway, or AWS EC2**.

CHAPTER 6

Software Requirement Specification

Frontend & Backend

Category	**Details**
Frontend	
Language/Framework	React.js with Tailwind CSS
Browser Compatibility	Latest versions of Chrome, Firefox, Edge, Safari
Dependencies	Axios, Redux (if needed), WebRTC, Socket.IO
IDE	VS Code (recommended)
Backend	
Language/Framework	Node.js with Express.js
Real-time Communication	Socket.IO
Authentication	JWT or OAuth 2.0
Database Interaction	Mongoose (for MongoDB)
APIs	RESTful APIs (user, session, file management)

Database & Hosting

Category	**Details**
Database	
Primary DB	MongoDB (hosted via MongoDB Atlas)
DevOps / Hosting	
Version Control	Git & GitHub
Containerization	Docker (optional for scalability)

Hosting - Frontend	Vercel / Netlify
Hosting - Backend	Render / Railway / AWS EC2
Hosting - Database	MongoDB Atlas
CI/CD	GitHub Actions

Communication Services & Tools

Category	**Details**
Communication Services	
Video/Voice Chat	WebRTC
Real-time Sync/Chat	Socket.IO
Other Tools	
Code Editor Integration	Monaco Editor (same as used in VS Code)
Linting/Formatting	ESLint, Prettier
API Documentation	Swagger

REFERENCES

[1] Feng, L., Yen, R., You, Y., Fan, M., Zhao, J., and Lu, Z. “CoPrompt: Supporting Prompt Sharing and Referring in Collaborative Natural Language Programming.” *arXiv*, vol. 2310.09235, arXiv, 2023, https://arxiv.org/abs/2310.09235.

[2] Chen, X., Li, S., Liu, S., Fowler, R., and Wang, X. “MeetScript: Designing Transcript-Based Interactions to Support Active Participation in Group Video Meetings.” *Proceedings of the ACM on Human-Computer Interaction*, vol. 7, no. CSCW2, Article 254, ACM, 2023, https://doi.org/10.1145/3610397.

[3] Dong, Y., Jiang, X., Jin, Z., and Li, G. “Self-Collaboration Code Generation via ChatGPT.” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM, 2023, https://doi.org/10.1145/3645669.

[4] Gaikwad, A., Agrawal, M., Goyal, J., Goyal, M., and Vinod, D. F. “A Collaborative Code Platform with Advanced AI Features and Real-Time Collaboration Tools.” *International Journal for Research in Applied Science and Engineering Technology*, vol. 12, no. 5, 1234–1242, IJRASET, 2024, https://doi.org/10.22214/ijraset.2024.

[5] Shi, L., Chen, X., Yang, Y., Jiang, H., Jiang, Z., Niu, N., and Wang, Q. “A First Look at Developers' Live Chat on Gitter.” *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21)*, ACM, 2021, pp. 573–584.

[6] Fathima, G., Kaif, M., and Nadeem, M. “CODEGEN: A React-Based Online Coding Workspace with Advanced Features and Real-Time Execution.” *International Journal of Computer Applications*, vol. 175, no. 3, pp. 25–31, Foundation of Computer Science, 2023, https://doi.org/10.5120/ijca2023922770.

[7] Verma, R., Nalisnick, E., and Naesseth, C. A. “Real-Time Collaborative Coding in a Web IDE.” *Proceedings of the 15th International Conference on Cooperative Design, Visualization and Engineering*, Springer, 2023, pp. 45–52.

[8] Kaur, M., Azrour, M., and Sinha, K. “Collaborative Virtual Reality Application for Interior Design.” *Proceedings of the 10th International Conference on Virtual Reality and Visualization*, IEEE, 2023, pp. 67–74.

[9] Wong, A. Y., Chekole, E. G., Ochoa, M., and Zhou, J. “On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies.” *Computers & Security*, vol. 128, article no. 103140, Elsevier, 2023, https://doi.org/10.1016/j.cose.2022.103140.

[10] Sinha, K., Swaminathan, K., Tambe, T., Zhang, J. J., and Buyuktosunoglu, A. “From Code to Container: Understanding the Importance of Container Images.” *Proceedings of the IEEE Symposium on Security and Privacy*, IEEE, 2024, pp. 112–125

.