Srishyla Educational Trust®, Bheemasamudra

GM INSTITUTE OF TECHNOLOGY

DAVANGERE.

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

7th Semester Project Work (21CSP76)

Synopsis on

"Collaborative Coding Platform"

Submitted in partial fulfillment of requirements for the award of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

4GM21CS095	**Sanjana V U**
4GM21CS098	**Shashank M N**
4GM21CS101	**Shreya Virupakshappa Katagi**
4GM21CS107	**Spoorthi Amaresh Kaddi**

Under the Guidance of

Dr. Arun Kumar B T
Assistant Professor
Department of CSE, GMIT

Visvesvaraya Technological University
Jnana Sangama
Belagavi-590018, Karnataka

ACADEMIC YEAR 2024-2025

CONTENTS

Chapter 1

INTRODUCTION

In the fast-paced world of software development, effective collaboration is essential for achieving successful project outcomes. While established platforms like GitHub provide robust version control and repository management, they often fall short in offering integrated features that facilitate real-time collaboration among developers. Although these tools are excellent for code management, they lack the ability to allow multiple users to simultaneously interact with the codebase in real-time. This absence of real-time collaboration results in developers having to rely on manual pull requests, branch reviews, and conflict resolution, which can introduce delays and create friction during the development process.

Furthermore, existing communication tools, such as Google Meet and Zoom, are not optimized for development workflows. They do not support coding directly within their interfaces, forcing developers to switch between different platforms to code, communicate, and manage projects. This constant context switching leads to fragmented workflows and increased miscommunication, which can cause inefficiencies and reduced productivity.

In addition, traditional project management platforms like Jira or Trello, though effective in managing tasks, do not integrate seamlessly with real-time coding environments. As a result, development teams often face the challenge of coordinating tasks, managing version control, and resolving issues while working across multiple disjointed tools. This disconnect can result in project delays, redundant work, and an overall decrease in the quality of the final product.

PROBLEM STATEMENT

This project aims to address these challenges by developing a comprehensive collaborative coding platform that seamlessly combines essential features such as project management, real-time code editing, integrated video conferencing, and robust version control into one unified interface. The platform will allow developers to collaborate on code in real-time, with multiple users able to edit, comment, and review the same codebase simultaneously, while also integrating powerful communication tools like video and voice chat to ensure that discussions and decisions can happen directly within the development environment.

Chapter 2

OBJECTIVES

- To Develop an intuitive and responsive user interface that simplifies navigation for administrators, collaborators, and viewers.
- To Implement live code editing, allowing multiple users to work on the same codebase simultaneously, minimizing conflicts.
- To Integrate video conferencing capabilities, enabling users to initiate calls directly within the platform for seamless communication.

Chapter 3

LITERATURE SURVEY

Sl. No.	Author(s)	Paper Name	Year	Description
1	Kumar, A., & Singh, P.	Enhancing Software Development Productivity through Real-Time Collaboration Tools	2020	Discusses the benefits of real-time collaboration tools in improving productivity and reducing errors in software development.
2	Smith, J.	Analyzing the Integration of Code Collaboration Features in Modern Development Platforms	2021	Highlights the limitations of current platforms like GitHub in providing real-time collaboration and the need for a unified solution.
3	Johnson, R.	Advances in Web Technologies for Real-Time Collaboration in Software Development	2022	Explores how technologies like WebSockets and cloud-based services enable real-time collaboration in software development.
4	Jones, T., & Taylor, S.	The Impact of Live Coding Environments on Learning and Engagement	2022	Investigates the positive effects of live coding environments on student engagement and learning in educational settings.

Chapter 4

ARCHIECTURE

The proposed system architecture is designed to follow a microservices model, ensuring modularity, scalability, and maintainability. By breaking down the platform into discrete, independently deployable services, each responsible for a specific functionality, this architecture ensures flexibility in scaling, adapting to evolving requirements, and easier maintenance over time.

System Architecture Overview

The architecture will adopt a microservices approach, with each service dedicated to a specific aspect of the platform, including user management, real-time collaboration, project management, and media services. This allows for independent deployment, scaling, and management of the platform's components, enabling better resource utilization and the ability to address varying demands effectively.

Frontend (React-Based User Interface)

The frontend of the platform will be developed using React, a powerful JavaScript library known for its efficient state management, component-based architecture, and support for single-page applications (SPAs). This choice will enhance the user experience by providing a responsive, dynamic interface that can handle real-time updates seamlessly.

Key Features:

1. Real-Time Updates: WebSockets will be used to enable live, bi-directional communication between the client and server. This will allow real-time code editing, chat, and project updates. Changes made by one user will be instantly reflected in all other connected clients, ensuring a fluid collaborative experience.

2. RESTful Communication: In addition to WebSockets, REST APIs will handle standard user requests such as user registration, authentication, project creation, and file uploads. The REST API will be optimized to minimize latency and ensure a smooth user experience.

3. User Authentication: The frontend will securely handle authentication via JWT (JSON Web Tokens), ensuring secure session management and access control.

Backend (Spring Boot for Microservices Architecture)

The backend will be built using Spring Boot, a highly efficient Java framework that simplifies the development of microservices and REST APIs. Spring Boot provides robust out-of-the-box support for security, database management, and integration with other services, making it an ideal choice for this architecture.

Database (PostgreSQL or MongoDB)

The backend will interact with either PostgreSQL or MongoDB, depending on the nature of the data being handled. PostgreSQL is a relational database ideal for structured data, while MongoDB is a NoSQL option better suited for unstructured or evolving data schemas.

Real-Time Collaboration (WebSockets)

WebSockets will serve as the backbone for real-time communication within the platform, enabling live collaboration on code, instant messaging, and real-time updates.

Video Conferencing (Twilio or Jitsi Integration)

Real-time video conferencing will be integrated into the platform using Twilio or Jitsi SDKs, enabling users to conduct face-to-face meetings or audio calls without leaving the platform..

Security Considerations

To ensure the platform’s security, several measures will be implemented, including end-to-end encryption for WebSocket communication, secure API endpoints protected by JWT, and role-based access control for managing user permissions. Furthermore, all sensitive data will be encrypted at rest and in transit, and third-party libraries will be regularly audited for vulnerabilities.

Containerization and Deployment

The platform will be containerized using Docker, allowing for consistent deployment across different environments. Kubernetes will be used for orchestration, automating the deployment, scaling, and management of containerized applications.

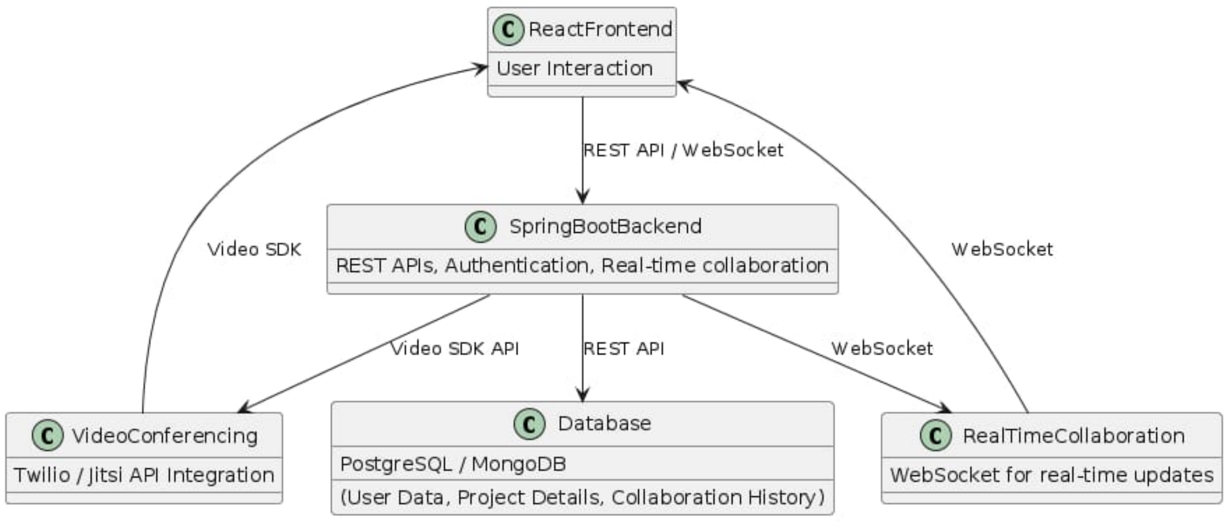

Fig : Flowchart of System Architecture

Methodology

Agile Development:

The project will adopt Agile methodologies to enable iterative development. This approach allows for flexibility and frequent adjustments based on user feedback, ensuring that the platform evolves to meet user needs.

User-Centered Design:

Throughout the development process, user research and usability testing will be conducted to ensure that the platform meets user needs effectively. Engaging with potential users will help identify pain points and areas for improvement.

Continuous Integration/Continuous Deployment (CI/CD):

CI/CD pipelines will be implemented for automated testing, integration, and deployment. This will ensure that the platform remains stable and up-to-date, allowing for quick iteration and feature enhancements.

Modular Development:

The project will be divided into distinct modules (e.g., user management, project management, collaboration tools) to enhance focus and facilitate parallel development efforts. This modular approach will streamline the development process and improve maintainability.

Chapter 5

SOFTWARE/HARDWARE REQUIREMENTS

Software requirements

Frontend:

- Node.js
- React.js
- Redux Toolkit for state management
- Socket.IO for real-time communication
- Axios for HTTP requests
- Tailwind CSS for styling
- Twilio SDK or Jitsi API for video conferencing

Backend:

- Java Development Kit (JDK)
- Spring Boot framework
- Spring Security for authentication and authorization
- PostgreSQL or MongoDB database
- Maven or Gradle for dependency management

Development Tools:

- Git for version control
- Docker for containerization and environment management
- Postman for API testing
- Integrated Development Environments (IDEs) like IntelliJ IDEA and Visual Studio Code

➢ **Hardware Requirements**

Development Machine:

- Minimum: 8 GB RAM, Dual-Core CPU, 256 GB SSD
- Recommended: 16 GB RAM, Quad-Core CPU, 512 GB SSD

Server:

Minimum: 4 GB RAM, 2 vCPUs, 20 GB Storage (for development and testing)

Recommended: 8 GB RAM, 4 vCPUs, 50 GB Storage (for production deployment)

Chapter 6
APPLICATIONS

The collaborative coding platform will have numerous applications across various sectors:

Educational Institutions:

The platform can support coding classes and collaborative projects, enabling students to work together in real time, enhancing their learning experiences.

Software Development Teams:

Development teams can benefit from integrated tools that streamline coding, communication, and project management, improving overall efficiency.

Open-Source Projects:

The platform will provide a dedicated space for contributors to manage issues, review code, and collaborate on open-source initiatives.

Remote Work Environments:

By combining coding, video conferencing, and project management tools, the platform will facilitate remote teams in achieving seamless collaboration.

Hackathons and Coding Competitions:

The platform can serve as an ideal environment for participants to collaborate, innovate, and showcase their coding skills during events.

REFERENCES

- Kumar, A., & Singh, P. (2020). Enhancing Software Development Productivity through Real-Time Collaboration Tools. Journal of Software Engineering, 15(4), 212-220.
- Smith, J. (2021). Analyzing the Integration of Code Collaboration Features in Modern Development Platforms. International Journal of Computer Science, 12(1), 45-58.
- Johnson, R. (2022). Advances in Web Technologies for Real-Time Collaboration in Software Development. Journal of Web Engineering, 20(3), 180-195.
- Jones, T., & Taylor, S. (2022). The Impact of Live Coding Environments on Learning and Engagement. International Journal of Educational Technology, 18(2), 110-125.
- O'Reilly Media - Collaboration in Software Development: Best Practices (2020)
- GitHub - GitHub Documentation on Code Collaboration (2021)

  https://docs.github.com/en/pull-requests/collaborating-with-pull-requests
- JetBrains - Real-Time Code Collaboration: Code With Me (2021)

  https://www.jetbrains.com/code-with-me/