Section A-Research paper
ISSN 2063-5346

Collaborative Application Development Tool

Prof. Prajakta Pawar[1], Prof. Yogesh Kadam[2], Tushar Rokade[3], Saurabh Bhalerao[4], Adnan Khan[5], Nikhil Kumar[6]

1,2,3,4,5,6 Bharati Vidyapeeth's College of Engineering, Lavale, Pune, India.

Email: pawar.prajakta@bharatividyapeeth.edu

Abstract

With the internet expanding rapidly, a growing number of applications that were previously desktop-based have transitioned to the web. Now, people can easily access applications anywhere and anytime using the internet. To develop these applications rapidly and efficiently, developers require tools such as code editors having effective development environment. The process of creating software applications or systems through the joint effort of multiple individuals or teams is referred as Collaborative application development. This emphasizes communication, coordination and cooperation among team members to build a high-quality application. The developers share tasks, exchange ideas, and contribute their skills and expertise to deliver a final product. The integrated development environment by using web socket technology allows users to work together on a project in a real time. Most of the existing collaborative application development uses code editor applications, distributing questionnaires, and conducting literature reviews. The true collaborative application development must offer a workspace for writing codes, executing codes, display results along with real-time collaboration, chat or community options and terminal building etc. This research work is focused on the development of such collaborative application. Thus, collaborative application development empowers developers work together to design, develop, test as well as deploy an application.

Keywords – Collaborative development, Integrated development environment, Realtime application development, Programming tools.

Introduction:

Earlier during 1960s or 1970s, the software development was often done by individual programmers or small teams working in isolation. Limited communication and collaboration tools were available then and most development was done consecutively, with little interaction among team members. Later Version Control Systems and Subversion permitted developers to manage and collaborate on shared code repositories. Further Distributed Version Control Systems like Git developed, providing more flexibility and allowing seamless collaboration amongst distributed teams. This allowed developers to work independently and merge their changes easily. Further the open-source software played a significant role in shaping collaborative development practices. Throughout its history, collaborative application development has experienced significant changes, driven by technological advancements, agile

methodologies, and the need for efficient collaboration among development teams. The focus of software development has shifted from individual work to team-based collaboration, enabling faster development cycles and improved code quality in software development.

Formerly, developers used to share files by sending an entire copy through email or other internal communication channels. When another developer wants to edit the file, they download and edit the copy locally and send the edited copy to the interested group through a communication channel. During this process, collisions may occur, and the developers must manually resolve the changes made by the prior developer. This process is slow and time-consuming. Collaborative code editing tools solve this problem by allowing multiple developers to access and edit the file simultaneously. Pair programming, which involves two or more developers working on the same program, has become quite popular due to its benefits, such as increased productivity, improved code quality, and easier problem-solving.

When software engineers work in teams to develop software, they often face the issue of manually merging changes they made to a module, which conflicts with the changes made by another developer to the same or different module. This happens because developers edit code separately and coordinate their work via a software configuration management system or source control. Most developers use various platforms and Integrated Development Environments (IDEs) to write, compile, and run their code. However, with the increasing trend of software engineers working in teams to develop products, conflicts often arise when two or more engineers make changes to the same module, resulting in the need for manual merging of the changes. These developers edit the code separately and coordinate their work via a software configuration management system or a source control. To solve this issue, a Collaborative Real-Time Coding approach has been proposed, which allows valid changes to be viewed by others in real-time. However, intermediate changes that cause the code to fail to compile can block other engineers from making related changes to the entity being modified. While collaborative real-time editing systems have been studied for the past two decades, research has primarily focused on collaborative textual and graphical editing. This work addresses the challenges in designing a collaborative real-time coding system and highlights the key differences compared to collaborative plain text editing and graphical editing [1].

This research work refers to a Collaborative Real-Time Coding method to eliminate the need for manual merging of changes by allowing valid changes to be seen by other team members in real-time. If intermediate changes cause the code to not compile, other team members can be blocked from making changes related to the entity being modified while still allowing them to work on most of the system. Collaborative real-time editing systems have been studied for the past two decades, but this study focuses on designing a collaborative real-time coding system.

Overview of collaborative working environment:

The software plays an important role to solve daily life problems. There has been a concerted effort to standardize and improve the software development process. This is primarily because of the increasing complexity, size, and distribution of software development projects, which go beyond what any single person or component can handle. Therefore, a standardized collaborative approach is necessary, involving a diverse range of people, skills, activities, processes, locations,

ISSN 2063-5346

tools, environments, configurations, and specifications. Software development is a collaborative process that involves divergent and convergent activities, carried out by individuals or teams, within a specific environment, towards achieving a set of objectives or outcomes.

Software development process involves set of practices that provide guidelines for designing, developing, testing, deploying, maintaining, and managing software. This process includes interactions between different parts that work together towards a common goal. A software development model is a high-level abstraction that embodies the entire development lifecycle. It facilitates and guides a set of tasks or activities to transform problem definitions and requirements into software. Various software development models have been adapted as different methodologies, aimed at standardizing and improving the process of software development. The table below presents a cross-section summary of some common software development models, including their related characteristics, advantages, and challenges. It serves as a baseline review template for relative comparisons, considerations, and reconciliations or possible combinations. Although there are inherent similarities among the models, such as reliance on collaborative development processes, accountability of the team and process along the lines of responsibility, roles, and functions, iteration within the process geared towards management of change, risks, and performance, and design, development, and testing activities aimed at achieving a common overall outcome.

Despite their unique features, software development models share some inherent similarities, such as the reliance on the collaborative development process and the team, accountability of the team and process along the lines of responsibility, roles, and functions, iteration within the process geared towards management of change, risks, and performance, etcetera. When implemented correctly at different stages of the software development process, any of the models can deliver quality solutions. The stages of the software development process are not always fixed, and the boundaries of the stages are not always clearly defined or differentiated. The development process stages involve different activities, which are grouped into development models, with some overlapping and mixing towards achieving the outcome. A typical software development project involves a team of people from diverse cultures, skill sets, technical expertise, and technological/nontechnical viewpoints. They work together on different tasks or separately on complementary tasks at each stage of the process towards a common goal while ensuring communication through various tools or media. Efficient collaboration and management in the software development process are critical due to the important role of software in society and other factors such as an increase in the size, complexity, and distribution involved in software development projects, which have generated a lot of attention leading to attempts aimed at standardizing and improving the development process.

The idea of utilizing Cloud Computing to improve collaboration in different activities is gaining momentum and making significant progress in various industries and fields. The following table illustrates the landscape of current knowledge on successful cases of leveraging Cloud Computing capabilities for collaborative software development from various industries. However, it should be noted that this table is not comprehensive due to various reasons, such as unrecorded or unpublished work, closely guarded or undocumented industrial intellectual property, or experimental projects yet to be verified or validated. Highlighting and reviewing

these works would help identify gaps and emphasize the need for further research efforts. This is to address the inefficiencies and inconsistencies of traditional software development processes, align software development with current trends and changing business requirements, leverage new concepts and methods for optimal development processes, and achieve economies of scale and the Most current Cloud-based solutions offered in industry offer more support for the coding and deployment stages of the software process and less for other stages, such as the requirements gathering, testing, and design stages. Some solutions attempt to integrate social communication tools to enhance communication, but this does not necessarily make the development environment a collaborative Cloud-based development platform. Enabling or enhancing collaboration in Cloud development environments by integrating social communication tools is one approach, but creating a fully collaborative development environment in the Cloud requires more than that [2].

Review of Literature:

Collaborative editing systems have been studied for the past 20 years, with most research concentrating on collaborative textual and graphical editing. Although there is no existing work in the field of collaborative real-time coding systems, we drew inspiration from the foundations of collaborative text editing. In 1989, Ellis and Gibbs proposed the operational transformation (OT) framework for concurrency management in distributed groupware systems, which addressed the challenges of having a real-time, highly reactive, concurrent editing environment for plain text. The OT framework transforms arriving operations against independent operations from the log to ensure convergence of document states. However, the correctness of the algorithm was challenged by the dOPT puzzle, which describes a scenario where clients diverge by more than one step in their state, breaking the algorithm's correctness.

In the field of collaborative editing systems, Jupiter is a system that implemented a centralized architecture, which is different from most other collaborative editing systems. Jupiter used a central server to mediate between any two clients, allowing only 2-way communication at any given time. The server was responsible for propagating changes to all other clients. When the central server received an operation request, it transformed the request if necessary, applied it to the local document state, and propagated it to other clients. This centralized approach relieved Jupiter of both the dOPT puzzle and precedence issues. Since only a 2-way synchronization took place at any given time, Jupiter also alleviated the issue of preserving precedence between operations [1].

Existing system for collaborative development:

At present following tools are being used for collaborative application development.

1. Saros

Several collaborative code editors currently in use employ a client-server architecture where a server holds a persistent copy of the shared document and each client has a local copy. One such example is Saros, which is an Eclipse plugin that enables real-time collaboration between two or more programmers by sharing changes. Saros is written in Java and uses XMPP for communication. It follows a client-server model for sending message updates to other clients and

ISSN 2063-5346

a peer-to-peer model for file transfer to enhance performance. In the event of direct connection failure, Saros uses a client-server mode for file transfer [9].

2. VS code:

Visual Studio code is an integrated development environment that offers several features and extensions that facilitate collaborative application development. The collaborative features and extensions make VS Code a powerful tool for team-based application development. VS Code enhances productivity and fosters efficient collaboration among developers working on the same project by enabling real-time collaboration, code sharing, version control integration, and communication

3. Github:

Collaborative application development and Git go hand in hand as Git is a widely used version control system that greatly facilitates collaboration among developers. Github is a platform where user or developer can post their project files in the form of repository. Repository can be public or private. It is one of the largest platforms for repository collaboration [10].

Proposed System for collaborative application development:

The following diagram shows the overall design of the system developed in this research work.

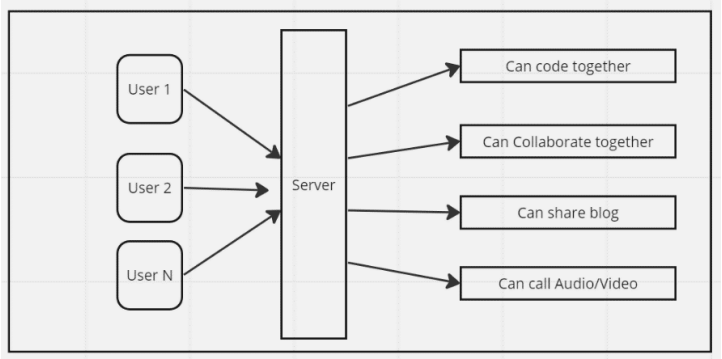


Fig-1: Overall design of the system

1. User Class:

Users can be different team members of the organization. They can be programmers, tester and any other crew member who deals with development of the product. Different users are managed by the classes and object of the constraints.

2. Admin Class:

Admin is the person who can control, manages and facilitate every task ad and every single feature of the end application. Admin will be the central authority. Admin can add or remove any other users from the system.

3. Assumptions and dependencies:

System will allow only intended and authorize entity to enter into the system and make use of it. Different users have different type of the functionality System settings cannot be changed by the any entity other than the central admin. Concurrent access allow only when entity sign in into the application. When any entity tries to communicate improperly or without any official credentials then it will be not allowed to make use of the software application. user have poor internet connection then it will only able to access the emergency feeder data. Poor internet connection may create the problem to use the system.

4. Functional use cases developed are as follows:
    - Admin Use case:

Admin will be the central authority in system. Admin can add and remove the programmers and other team members. Admin can manage the flow of the application and behavior of the different entities involved in the process.

- Programmer use case:

Programmers are allowed to develop application in concurrent way Programmers will use the Jupyter notebook cloud-based platform for collaborative coding Real time communication is possible Error solving and bug fixing will be done at very less time cost.

- Code room space:

Code room space is the area where different user can create a shared space. In this section different users or programmers are allowed to write a code in same space where they can see each other's work. They can suggest improvement or marks error while coding.

- Chat room space:

Admin is the person who can control, manages and facilitate every task and every single feature of the end application. Admin will be the central authority. Admin can add or remove any other users from the system

- Blog space:

Blog space is nothing but the platform like stack overflow where different users can be able to post new ideas, doubt, and the any other information Other users are able to like comment and share that post Blog space will help in the increased in productivity.

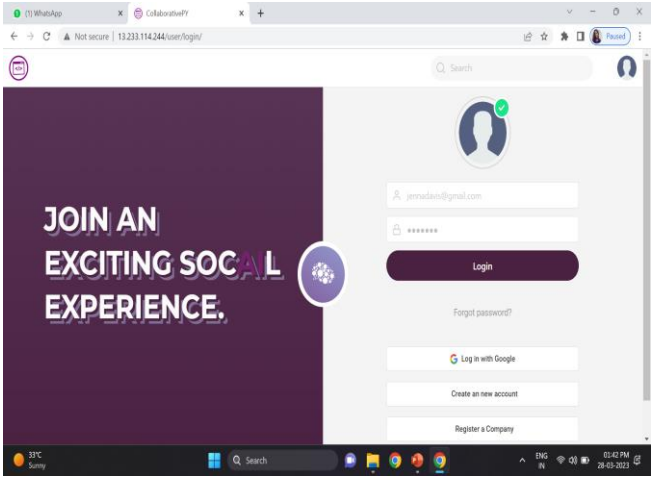
- Online meeting system:

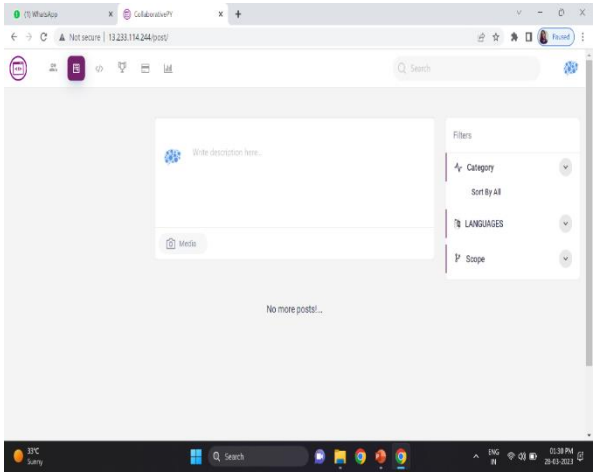
Users are able to communicate with each other with the help of the features like audio call and video call through web socket-based communication.

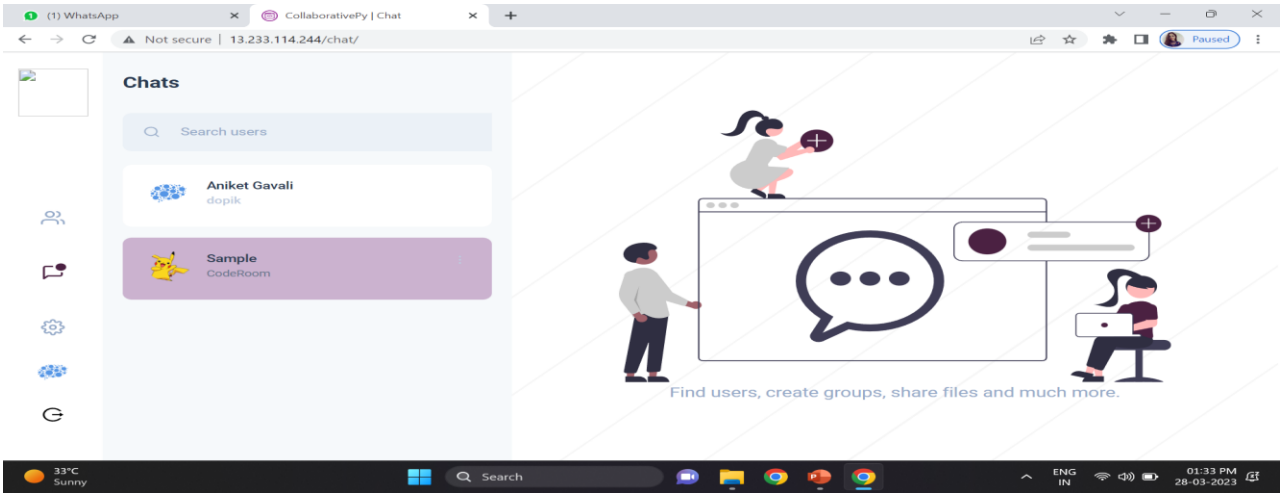
With the help of the feature like chat room and the code room, the problems which can be occurs during the product development will be solved easily which results in better development of applications.

Proposed System Implementation:

Collaborative application development platform developed in this research work looks as follows.







Thus, collaborative application development allows effective communication, increased debugging and conferencing for real-time collaboration among team members.

Conclusion and Future Work:

This research work presents the real-time collaboration application development, including its design and implementation. This is a web-based platform that assists programmers in executing and visualizing source code in real-time using a terminal. It also enables real-time collaboration with other programmers via chat and project invites, and facilitates project management, including the import and export of shared projects. It supports programming languages such as Python and its key features include workspace provision for executing and building source code, real-time collaboration, chat, and terminal building capabilities. This platform can make

availability of new powerful and effective tools for development environment. Collaboration of programmers while actual development of the product leads to the faster and economic benefits to the organization that helps in growth of the organization. In future many tools can be integrated with collaborative application development platform like docker or other tool to develop robust application and single space. Also, we can integrate the numerous programming languages.

References:

[1] "Collaborative Real Time Coding or How to Avoid the Dreaded Merge " -Stanislav Levin The Blavatnik School of Computer Science Tel Aviv University Tel-Aviv, Israel stanisl@post.tau.ac.il Amiram Yehudai The Blavatnik School of Computer Science Tel Aviv University Tel-Aviv, Israel amiramy@tau.ac.il

[2] "Cloud Based Collaborative Software Development: A Review, Gap Analysis and Future Directions" - Stanley Ewenike, Elhadj Benkhelifa and Claude Chibelushi School of Computing and Digital Technologies, Cloud Computing and Applications Research Lab Staffordshire University, Stoke of Trent, UK .

[3] "CodeR: Real-time Code Editor Application for Collaborative Programming"-Aditya Kurniawan, Aditya Kurniawan, Christine Soesanto, Joe Erik Carla Wijaya

[4] "Supporting distributed software development by modes of collaboration" T Schümmer, JM Haake - ECSCW 2001: Proceedings of the Seventh European Conference on Computer supported cooperative work Germany September 2001 – Springer

[5] Dewan, P. (2010). "Towards and Beyond Being There in Collaborative Software Development"

In: Mistrík, I., Grundy, J., Hoek, A., Whitehead, J. (eds) Collaborative Software Engineering. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-10294-3_7

[6] https://www.ibm.com/docs/en/ram/7.5.4.2?topic=management-collaborativedevelopment-operations

[7]https://developer.servicenow.com/dev.do#!/learn/courses/quebec/app_store_learnv2_devenvironment_quebec_managing_the_development_environment/app_store_learnv2_devenvironment_quebec_source_control/app_store_learnv2_devenvironment_quebec_collaborative_development

[8]https://docs.mendix.com/refguide/collaborative-development

[9]https://marketplace.eclipse.org/content/saros-distributed-collaborative-editing-and-pair-programming

[10] https://www.techtarget.com/searchitoperations/definition/GitHub