RESEARCH ARTICLE OPEN ACCESS

CodeDocs: AI-Powered Real-Time Collaborative Coding with Automated Documentation

Prof. Mangesh Devkate, Omkar Daptare, Anurag Ghortale, Sarthak Bobhate, Aniket Agvane

Department of Information Technology, Zeal College of Engineering and Research, Pune, India

ABSTRACT

This paper presents CodeDocs, an AI-powered real-time collaborative coding platform with automated documentation generation. Documentation is a critical yet often neglected aspect of software development. Developers tend to focus heavily on writing code, leaving documentation as an afterthought, which leads to poor maintainability and understanding of codebases. To address this, CodeDocs was developed as a web-based system that seamlessly integrates coding, collaboration, and documentation into a single platform. Leveraging Artificial Intelligence (AI) and Natural Language Processing (NLP), the system automatically generates human-readable documentation as code is written. The platform also supports real-time collaboration through shared coding rooms. A multi-tiered architecture comprising a React.js frontend, Node.js/Express.js backend, Python/Flask documentation engine, and MongoDB database was employed. The system supports customizable documentation prompts and structured output, making it suitable for students, developers, and professional teams alike.

Keywords — Artificial Intelligence (AI), Natural Language Processing (NLP), Code Documentation, Collaborative Coding, Generative AI, Software Development, Web Application, Data Management.

I. INTRODUCTION

The necessity of documentation in software development is universally acknowledged, yet in practice, it is routinely neglected. Programmers invest their efforts in writing functional code while documentation is deferred or skipped entirely, resulting in codebases that are difficult to understand, maintain, or extend.

This challenge intensifies in team environments. Without clear documentation, both existing and new developers struggle to comprehend the logic behind a system. Several automated documentation tools have been developed, but they produce simplistic descriptions and lack real-time user interaction.

Modern Artificial Intelligence (AI) advancements enable the generation of meaningful, contextually accurate code explanations. Considering this, CodeDocs was developed as a unified platform that integrates coding, real-time collaboration, and AI-driven documentation generation. Users can write code, collaborate in shared rooms, and produce documentation simultaneously, with customizable output prompts to suit individual needs.

The system is built on a modular, multi-tiered architecture ensuring stability, scalability, and ease of maintenance. The primary goal of CodeDocs is to minimize manual documentation effort while maximizing code comprehensibility and developer productivity.

II. IMPORTANCE OF TECHNOLOGY

Reduction of Manual Documentation Efforts: Automating documentation significantly reduces the burden on software engineers, freeing them to focus on core development tasks rather than writing explanations manually .

Improvement of Coding and Debugging Efficiency: When developers are not occupied with writing documentation manually, they can dedicate more time to writing clean, efficient code and resolving defects.

Collaboration: The platform enables developers to work together in real time, facilitating communication and simultaneous code editing within a shared environment, thus reducing coordination overhead.

Code Logic Understanding: The AI-powered explanation tool assists software engineers, especially in large-scale projects, by making the logic behind complex code understandable at a glance.

Greater Accuracy of Documentation: Automated generation eliminates human errors, producing consistently formatted and accurate documentation free from omissions and inconsistencies.

Time Savings in Development Process: Documentation produced automatically frees valuable developer time that can be redirected toward achieving project milestones.

Learning and Training Purpose: The code explanation feature serves as a valuable learning resource for students and junior developers building their understanding of coding concepts.

System Scalability: The system is designed to handle increasing numbers of users and growing volumes of data while maintaining performance and responsiveness.

III. LITERATURE SURVEY AND RELATED WORK

Several studies in automated code documentation and collaborative development environments have laid the foundation for this research. As software complexity grows, efficient documentation and team coordination have become increasingly critical.

Feng et al. introduced CodeBERT, a transformer-based model capable of understanding both programming and natural languages. The model generates semantically accurate code descriptions and outperforms rule-based systems. However, CodeBERT operates offline and cannot be embedded in a live development environment.

Ahmad et al. proposed a deep learning approach using neural machine translation for automatic code documentation. While effective in producing documentation, their work did not address real-time collaboration.

Goldman et al. presented Collabode, a browser-based collaborative programming environment supporting real-time code editing. Although it enhances team interaction, it lacks any intelligent documentation capability.

Sun et al. developed a web-based real-time synchronization tool enabling collaborative coding. While effective for multi-user editing, it does not support automated document generation.

Fluri et al. studied traditional documentation systems such as Doxygen and Javadoc, which extract documentation from inline code comments. Despite their widespread use, they require significant manual effort and fail to understand code logic automatically.

From the literature, it is evident that existing systems focus either on AI-based documentation or collaborative coding, but rarely combine both, motivating the development of CodeDocs.

A. Research Gap

AI-based models such as CodeBERT operate offline and cannot integrate with live development workflows. Collaboration tools like Collabode facilitate code sharing but lack intelligent documentation. Existing tools also do not offer customizable output explanations tailored to individual user needs. Furthermore, documentation and code writing are disconnected processes in current systems. CodeDocs addresses all these gaps simultaneously by providing an integrated, real-time, AI-powered, and customizable platform.

TABLE I. Comparative Analysis of Existing Systems and Proposed System

Study	Automated Documentation	AI-Based Processing	Real-Time Collaboration	Custom Output
Feng et al.(1)	Yes	Yes	No	No
Ahmad et al.(2)	Yes	Yes	No	No
Goldman et al.(3)	No	No	Yes	No
Proposed System	Yes	Yes	Yes	Yes

A number of methods for code documentation and cooperative coding have been proposed in recent years. Each of them sought to increase the efficiency of the development process by reducing the level of human labor, making the code more legible and improving the cooperation in its creation.

The research of Feng et al. proposes a model called CodeBERT that can be used not only for programming but also in natural language processing tasks. Having knowledge about the context of the code, CodeBERT can create summaries and descriptions for it. It performed very well when conducting code search and generating code documents. Nevertheless, this model can operate only in an offline mode and is not applicable during the process of coding.

Another example of machine learning applied to coding was provided by Ahmad et al. who managed to create a system that translate code into natural language. In such a way, it became clear how the deep learning method can be used in order to perform the operation of code translation. This solution certainly improves the comprehensibility of the code. In the field of collaboration, we have the research conducted by Goldman et al., who developed Collabode as a collaborative environment intended for collective writing, reading, and modification of programming code. Collaboration in this case will be improved, as it will involve communication among users. Nevertheless, there cannot be any intelligent process for creation of any documentations or comments concerning programs in this application, as it can only be performed manually.

As for the paper presented by Sun et al., it considers development of a real-time collaborative editing system using operational transformations and other methods of synchronization. Using such technologies, one can deal with consistency problems occurring when several users are working simultaneously on the same document, thus solving a rather significant issue. At the same time, one cannot say that this solution is suitable for programming, as it does not consider any analysis or explanation of program code or documentation.

Traditional methods of document generation can be regarded as Doxygen and Javadoc. The basic concept of both programs is that it is possible to create documents on the basis of comments written within the program code. Despite its convenience, this approach cannot be called fully automated. It can be said that recent developments within the field of artificial intelligence have proven to be promising in terms of making improvements in the quality of documentation using AI-enabled tools that are able to interpret the semantics of the codes. However, the problem lies in the fact that most such

systems operate as standalone programs and do not incorporate collaboration through interaction.

From the discussion above, it can be seen that existing systems either focus on creating intelligent documentation or working on real-time collaboration, but rarely do they perform both functions simultaneously. Another common problem is the lack of customization or real-time collaboration when writing codes. In view of this problem, it can be seen that there is need for an integrated solution, which is exactly what CodeDocs provides.

IV. METHODOLOGY

A. System Overview

CodeDocs is implemented as a web-based application that merges coding, real-time collaboration, and automatic documentation generation into one unified platform. Its primary goal is to streamline software development by reducing the time spent on documentation while improving code understandability.

The system is divided into three primary modules: the Frontend Module, the Backend Module, and the Documentation Processing Module. Each module handles a specific function and communicates with others to deliver full end-to-end functionality.

The Frontend Module provides the user interface, a web page with an integrated code editor where users can write code, create rooms, and collaborate with other developers.

The Backend Module controls authentication, session management, room tracking, request routing, and inter-module communication. It relays code inputs to the documentation engine and synchronizes updates across participants in real time.

The Documentation Processing Module is the core intelligence of the system. It analyzes submitted code using AI algorithms and generates structured documentation reports explaining code functions, logic, and parameters. Customizable prompt inputs allow users to tailor documentation style and depth.

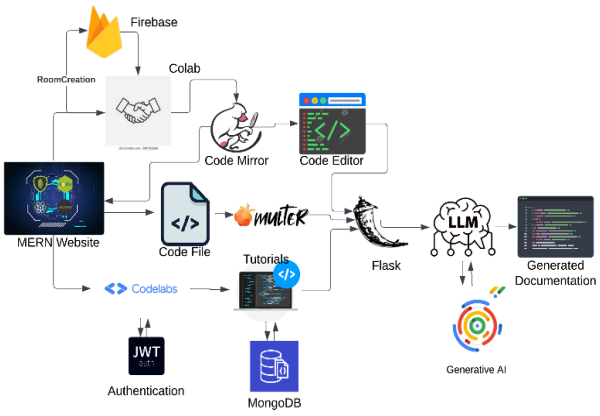

Fig. 1. System Architecture of CodeDocs

B. Workflow

The CodeDocs workflow describes the step-by-step interaction between system components to deliver an integrated coding, collaboration, and documentation experience.

1. Login into the System: Users log into CodeDocs via a web browser and enter a room ID to create or join a coding session.

2. Creation of Collaborative Environment: Upon joining, a collaborative session is established at the backend and tracked across all participants.

3. Code Input and Real-Time Editing: Users write or edit code in the integrated editor. All changes are reflected in real time across all room participants.

4. Data Transmission to Backend: Code changes on the frontend are transmitted to the backend via API calls for processing and documentation generation.

5. Backend Processing and Request Handling: The backend matches requests to their respective sessions and processes multiple requests simultaneously.

6. Documentation Generation: The documentation module analyzes submitted code, identifies elements such as functions and variables, and generates contextual documentation. User-defined prompts further customize the output.

7. Result Exporting: Generated documentation is delivered from the documentation module through the backend to the frontend for the user to view.

8. Real-Time Data Synchronization: Code and documentation updates are synchronized continuously across all participants to ensure consistency.

9. Continuation: The process iterates continuously as long as participants remain active, triggering documentation regeneration on every code change.

V. IMPLEMENTATION

A. Frontend Implementation

The frontend was developed using React.js with HTML and CSS, creating an efficient and user-friendly interface. It features an integrated code editor and supports collaborative room creation, session joining, and viewing of auto-generated documentation. The frontend communicates with the backend via API calls to transmit code and receive documentation responses in real time.

B. Backend Implementation

The backend was built using Node.js and Express.js to handle user requests, manage sessions, and coordinate communication between frontend and documentation modules. It manages authentication, room creation, request routing, and real-time updates. All participants in a session receive

synchronized code changes instantly through the backend layer.

C. Documentation Engine Implementation

The documentation engine was developed using Python and Flask. It uses AI technologies to analyze source code, understand its logic, and generate human-readable explanations. It identifies key elements such as functions, parameters, and variables, and documents them appropriately. Customizable prompts allow users to define the depth and style of documentation output.

D. Real-Time Collaboration Implementation

Real-time collaboration allows multiple users to co-edit code simultaneously. Every change made by any user is propagated to all participants in the room almost instantaneously through continuous frontend-backend communication.

E. Technologies Used

CodeDocs was developed using the following technologies:

- Frontend: React.js, HTML, CSS
- Backend: Node.js, Express.js
- Documentation Engine: Python, Flask
- Database: MongoDB
- Authentication: Firebase
- Version Control: Git, GitHub

F. Integration of System Components

The frontend, backend, and documentation module are integrated into one cohesive system. The frontend communicates with the backend via REST APIs, while the backend interfaces with the documentation engine through dedicated API calls. This layered integration ensures a clean separation of concerns while maintaining seamless end-to-end data flow.

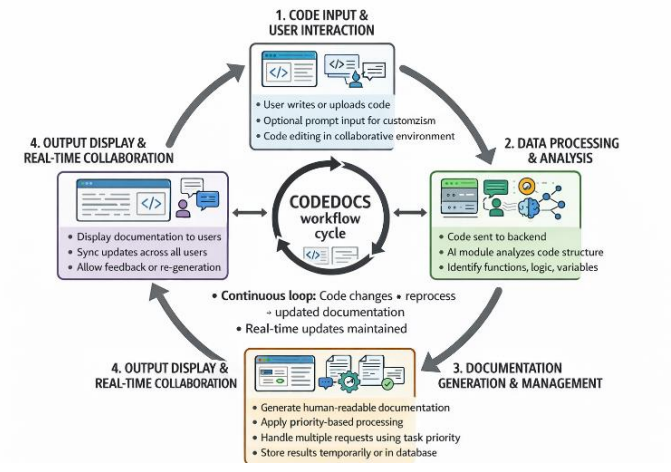

Fig. 2. System Implementation Results

VI. CONCLUSION

CodeDocs demonstrates itself as an effective and practical solution for improving coding practices and collaboration in software development. The system successfully combines AI-powered automatic documentation with real-time collaborative coding in a single integrated platform.

A key achievement is the ability to automatically generate human-readable documentation from source code, reducing developer effort and making code understanding faster and more accessible. The collaborative environment ensures teams can work efficiently in real time with all members staying synchronized.

Customizable documentation prompts and structured output make CodeDocs applicable across diverse user groups, from students learning to code to professional developers on large-scale projects. The modular architecture also ensures scalability and ease of maintenance.

In conclusion, CodeDocs effectively bridges the gap between code writing, team collaboration, and intelligent documentation, offering a unified and efficient environment for modern software development workflows.

ACKNOWLEDGMENT

We want to say thank you to the Department of Information Technology at Zeal College of Engineering and Research Pune, India for giving us the things we needed like infrastructure, resources and constant help for this research project. We are also very thankful to Prof. Mangesh Devkate and, to all the faculty members and friends who gave us good technical advice, ideas and comments while we were working on CodeDocs.

REFERENCES

[1] Z. Feng et al., "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," EMNLP Findings, 2020

[2] W. Ahmad et al., "A Transformer-based Approach for Source Code Summarization," ACL, 2020.

[3] M. Goldman, G. Little, and R. Miller, "Collabode: Collaborative Coding in the Browser," CHI Conference, 2011.

[4] D. Sun et al., "Operational Transformation for Real-Time Collaborative Editing Systems," CSCW, 2013.

[5] B. Fluri et al., "Change Distilling: Tree Differencing for Fine-Grained Source Code Change Extraction," IEEE Transactions on Software Engineering, 2007.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.

[7] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.

[8] T. Mens and S. Demeyer, Software Evolution, Springer, 2008.

[9] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 2005.

[10] Flask Documentation, "Flask Web Framework."

[11] Node.js Foundation, "Node.js Documentation."

[12] Meta Platforms Inc., "React.js Documentation."

[13] MongoDB Inc., "MongoDB Documentation."

[14] A. LeClair, S. Jiang, and C. McMillan, "A Neural Model forGenerating Natural Language Summaries of Program Subroutines."

[15] Automatic Source Code Summarization with Graph Neural Network Code Summarization, Research group in software engineering and machine learning.

[16] Title: Towards Automatically Generating Summary Comments – Classic Research on Automatic Comment Generation.