International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Self-Hosted AI Coding Assistant for Secure Real-Time Code Generation

[1]Hemant Vaidya, [2]Kartik Nayak, [3]Aditi Singh Thakur, [4]Prof. Sarita Khedikar

[1,2,3]Student, Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India
[4]Professor, Dept. of AI & ML, Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra, India

Abstract - **In the modern software development landscape, the demand for intelligent, secure, and efficient coding solutions has increased significantly. Traditional AI-based coding assistants primarily rely on cloud-based architectures, which introduce challenges such as data privacy risks, latency, and dependency on external services. These limitations restrict developers and organizations from adopting AI tools in secure or offline environments. To overcome these issues, there is a growing need for a self-hosted solution that ensures data control, flexibility, and real-time performance while maintaining high development efficiency.**

This research proposes a Self-Hosted AI Coding Assistant for Secure Real-Time Code Generation, designed to provide a privacy-focused and integrated development environment. The system enables developers to generate, edit, and execute code using locally hosted AI models, eliminating reliance on cloud APIs. It integrates features such as AI-driven code suggestions, chat-based assistance, sandboxed execution, and an interactive code editor. Built using modern technologies like Next.js, React, Tailwind CSS, Monaco Editor, Docker, WebContainers, and Ollama, the platform ensures scalability, performance, and security while enhancing overall developer productivity.

Keywords: Self-Hosted AI, Code Generation, AI Coding Assistant, Local LLM, Secure Development Environment, Real-Time Execution

I. INTRODUCTION

With the rapid advancement of artificial intelligence in software development, there is a growing demand for intelligent systems that can assist developers in writing, debugging, and managing code efficiently. Modern AI-powered coding tools have significantly improved productivity by enabling automated code generation and real-time assistance. However, most of these solutions rely heavily on cloud-based infrastructures, which introduce challenges such as data privacy concerns, increased latency, limited customization, and dependency on external services. These issues become critical for organizations handling sensitive data or requiring offline and secure development environments.

To address these limitations, this project proposes a **Self-Hosted AI Coding Assistant for Secure Real-Time Code Generation**, designed to provide a fully integrated and privacy-focused development platform. The system combines a web-based integrated development environment (IDE) with locally hosted AI models to enable real-time code generation, execution, and interaction without relying on external APIs. By leveraging modern technologies and secure sandbox environments, the platform ensures efficient performance, enhanced security, and greater control over data. This approach not only improves developer productivity but also provides a scalable and customizable solution suitable for both individual developers and enterprise-level applications.

1.1 Project Aims and Objectives

The development of a self-hosted AI coding assistant requires clearly defined goals to ensure security, efficiency, and scalability in modern software development environments. This project aims to overcome the limitations of cloud-based AI coding tools by providing a locally deployable solution that enhances data privacy, reduces latency, and offers greater control over system functionality. The platform integrates real- time code generation, execution, and AI-driven assistance into a unified development environment, enabling developers to work more efficiently and securely.

Objectives and Aims:

1. To develop a self-hosted AI coding assistant for secure code generation.
2. To enable real-time code generation and execution within a web-based IDE.
3. To integrate local large language models using Ollama for offline functionality.
4. To implement secure sandbox environments using containerization technologies.
5. To provide AI-powered chat assistance for debugging and code explanation.
6. To design an interactive and user-friendly interface using modern frontend technologies.
7. To ensure data privacy by eliminating dependency on external cloud APIs.
8. To support multiple programming languages and development frameworks.
9. To enhance developer productivity through automation and intelligent suggestions.

10. To build a scalable and modular system architecture for future expansion

1.2 System Objectives

The system is designed with a strong focus on performance, security, scalability, and usability to meet the evolving needs of modern developers. It ensures high performance by enabling real-time code generation and execution with minimal latency through locally hosted AI models. Security is a primary concern, achieved by processing all data within a controlled environment and utilizing sandboxed execution through containerization technologies. This prevents unauthorized access and ensures safe code execution.

The architecture is modular and extensible, allowing easy integration of new features such as additional AI models, plugins, and development tools without affecting existing functionalities. The system also emphasizes usability by providing an intuitive and responsive interface, enabling developers to interact seamlessly with the AI assistant. Furthermore, it supports efficient resource management and scalability, making it suitable for both individual users and enterprise-level deployment. Overall, the system objectives aim to deliver a secure, high-performance, and future-ready AI-powered coding environment

1.3 Background of Project

In recent years, the rapid growth of artificial intelligence and developer tools has transformed the way software is developed. AI-powered coding assistants have become increasingly popular due to their ability to automate repetitive tasks, generate code snippets, and assist in debugging. However, most existing solutions rely on cloud- based services, which require continuous internet connectivity and involve sharing sensitive code with external servers. This raises significant concerns regarding data security, privacy, and compliance, especially for organizations working with confidential information.

Additionally, cloud-based systems often introduce latency and limit customization, making them less suitable for developers who require fast, flexible, and offline-capable solutions. With the emergence of open-source large language models and local deployment frameworks, it has become feasible to build self-hosted AI systems that provide similar capabilities without compromising security. This project is motivated by the need to develop a secure, efficient, and locally controlled AI coding assistant that integrates code generation, execution, and intelligent assistance within a single platform. By leveraging modern technologies, the system aims to provide a reliable and scalable alternative to traditional cloud-based coding assistants.

II. COMPONENTS

The proposed Self-Hosted AI Coding Assistant system consists of multiple integrated components that work together to deliver a secure, scalable, and high-performance development environment. Each component is designed to handle specific functionalities such as user interaction, code processing, AI-based generation, and secure execution. The modular architecture ensures flexibility, maintainability, and efficient system performance.

2.1 Software components for processing the system

i) Frontend Layer

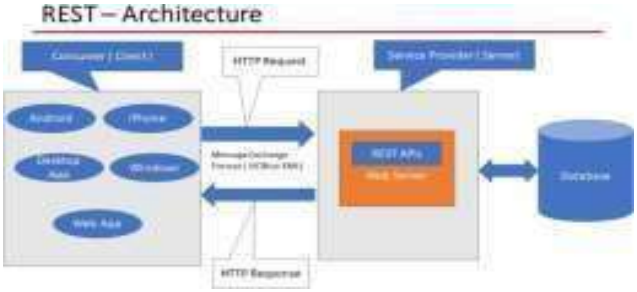
The frontend layer provides an interactive and user-friendly interface for developers to interact with the system. It is developed using modern technologies such as Next.js and React, ensuring fast rendering and seamless navigation. The user interface is designed using Tailwind CSS and ShadCN UI to provide a clean, responsive, and visually appealing experience.

This layer includes essential features such as the code editor interface, project dashboard, file explorer, and AI chat interface. It enables users to write, edit, and manage code efficiently while interacting with the AI assistant in real time. The frontend is optimized for performance and usability, ensuring smooth operation across different devices and environments.

ii) Backend Layer

The backend layer acts as the core processing unit of the system, managing all server-side logic and communication between components. It handles API requests, user authentication, project management, and interaction with AI models. The backend is designed using a scalable architecture that ensures efficient handling of multiple user requests simultaneously.

It also manages communication between the frontend, database, and AI engine, ensuring secure and reliable data processing. The modular structure allows easy integration of new features such as additional AI capabilities, plugins, and external tools without affecting existing functionalities.



iii) Database Layer

The database layer is responsible for storing and managing all system data in a structured and efficient manner. It includes user information, project files, code snippets, session data, and system configurations. Modern database technologies are used to ensure

fast data retrieval, consistency, and scalability.

The system is designed to handle large volumes of data while maintaining high performance. Proper indexing and optimized queries are implemented to ensure efficient data access and real-time synchronization across different components of the application

iv) AI Engine Module

The AI engine is the core component responsible for intelligent code generation and assistance. It utilizes locally hosted large language models through Ollama, enabling secure and offline AI processing. This eliminates dependency on external APIs and ensures complete data privacy.

The AI engine supports functionalities such as code generation, auto-completion, debugging assistance, and natural language interaction. It processes user prompts and generates relevant code snippets in real time, enhancing developer productivity and reducing manual effort.

v) Code Editor Module

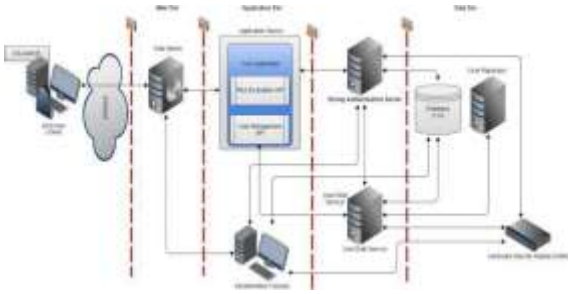
The code editor module provides a powerful and interactive coding environment within the browser. It is implemented using Monaco Editor, which offers features similar to modern IDEs such as syntax highlighting, auto-completion, formatting, and error detection.

This module allows users to write and edit code efficiently while receiving real-time suggestions from the AI engine. It also supports multiple programming languages, making it suitable for a wide range of development tasks.

vi) Authentication and Security Module

The authentication and security module ensures secure access to the system by implementing user authentication and authorization mechanisms. It manages login, session handling, and user permissions to prevent unauthorized access.

This module plays a crucial role in protecting sensitive data and maintaining system integrity. It ensures that only authorized users can access and modify projects, thereby enhancing overall platform security.



vii) Execution and Sandbox Module

The execution module enables real-time code execution in a secure and isolated environment. It uses technologies such as Docker and WebContainers to create sandboxed environments where code can run safely without affecting the host system.

This ensures that potentially harmful code does not compromise system security. It also allows users to test and preview their applications instantly, improving development speed and efficiency

III. METHODOLOGY

The development of the proposed Self-Hosted AI Coding Assistant follows a structured and systematic methodology to ensure efficiency, scalability, and reliability. The process consists of multiple stages including requirement analysis, system design, development, testing, deployment, and maintenance. Each phase is carefully planned to ensure that the system meets user requirements and performs effectively in real- world development environments.

1. Requirement Analysis

The initial phase involves gathering and analyzing system requirements based on developer needs and current limitations of existing AI coding tools. Key functionalities such as real-time code generation, AI-based assistance, secure execution, and local model integration are identified. Both functional and non-functional requirements, including performance, security, scalability, and usability, are defined to establish a clear foundation for system development.

2. System Design

In this phase, the overall system architecture is designed, including frontend, backend, database, and AI engine components. User interface layouts for the code editor, dashboard, and AI chat module are planned to ensure an intuitive user experience. Data flow between components and API structures are also defined. The design phase acts as a blueprint for the development process, ensuring proper integration of all modules.

3. Development

The development phase focuses on implementing the system using modern technologies such as Next.js and React for the frontend, along with backend services for handling logic and communication. Core features such as the AI-powered code generation engine, Monaco-based code editor, file management system, and AI chat assistant are developed. Integration of local AI models using Ollama is carried out to enable secure and

offline functionality.

4. Database Integration

This phase involves integrating the database with the backend to enable efficient data storage and retrieval. Data models are created for users, projects, files, and sessions. Proper indexing and optimization techniques are implemented to ensure fast query performance and real-time data synchronization across the application.

5. Testing

Testing is performed to ensure that the system functions correctly and meets quality standards. Various testing methods such as unit testing, integration testing, and system testing are conducted. The AI-generated outputs are evaluated for accuracy, and the execution environment is tested for security and reliability. Performance testing is also carried out to ensure the system can handle multiple users efficiently.

6. Deployment

In the deployment phase, the system is made available for users through a production environment. The application is deployed using modern hosting platforms or local servers, depending on the use case. Configuration of the database, AI models, and execution environment is completed to ensure smooth operation. This phase ensures that the system is accessible and ready for real-world usage.

7. Maintenance

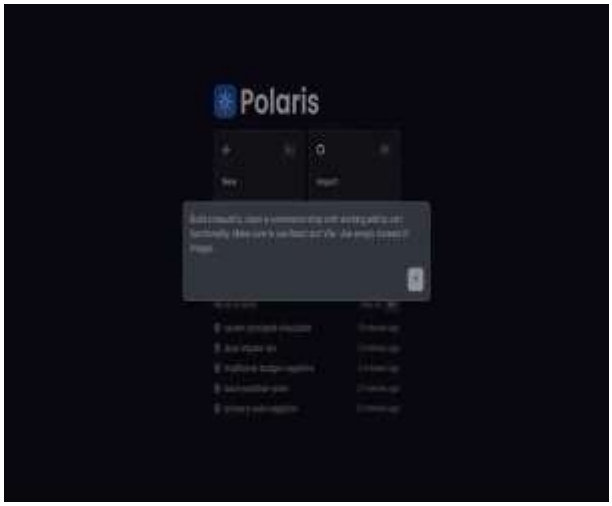
The final phase involves continuous monitoring and improvement of the system. Regular updates, bug fixes, and feature enhancements are implemented based on user feedback. System performance and security are continuously evaluated to ensure long-term reliability and adaptability to future technological advancements.

IV. RESULT

The developed Self-Hosted AI Coding Assistant successfully demonstrates an efficient and secure environment for real-time code generation and execution. The system integrates an AI-powered code generation engine with a web-based development interface, enabling users to write, edit, and execute code seamlessly. By utilizing locally hosted models, the platform ensures data privacy while maintaining consistent performance without dependency on external cloud services. The results indicate improved developer productivity through automation, faster code generation, and reduced manual effort in debugging and development tasks.

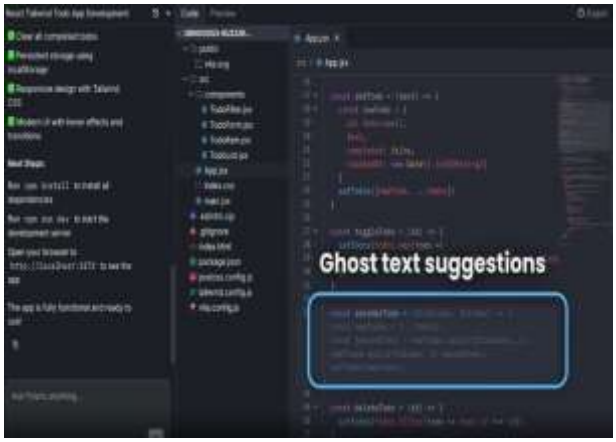
4.1 User Interaction and Workflow

The system provides a streamlined workflow where users can interact with the AI assistant through natural language prompts. Upon receiving input, the AI engine processes the request and generates relevant code in real time. Users can modify the generated code within the integrated editor and execute it instantly. This interactive workflow enhances efficiency and reduces development time significantly.



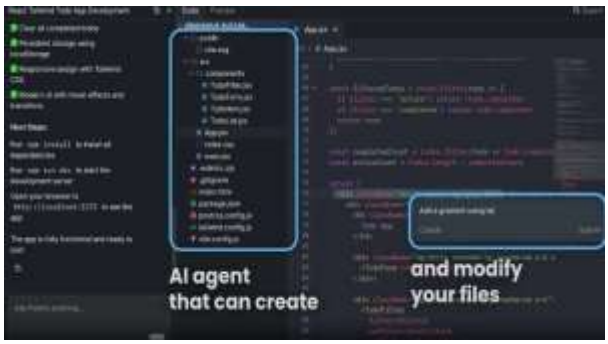
4.2 Code Editor and Development Environment

The integrated code editor offers features such as syntax highlighting, auto-completion, and real-time AI suggestions, creating a complete IDE-like experience within the browser. Users can manage files and projects through a structured interface, enabling organized development. The seamless interaction between the editor and AI assistant allows continuous support during coding tasks.



4.3 Real-Time Code Execution

The system enables secure execution of code using sandboxed environments. Users can run applications directly within the platform and view outputs instantly. This feature eliminates the need for external setup and allows developers to test and validate their code in real time, improving development speed and accuracy.

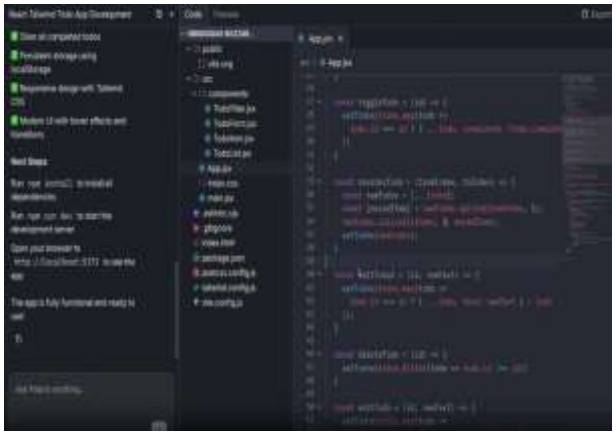


4.4 AI-Based Code Generation and Assistance

The AI engine generates context-aware code based on user prompts and provides suggestions for optimization and debugging. It also supports conversational interaction, allowing users to request explanations or modifications. The use of locally hosted models ensures that all processing is performed securely without exposing sensitive data.

4.5 System Performance and Security

The system demonstrates high performance with minimal latency due to local processing of AI models. Security is enhanced through sandboxed execution and controlled data flow, ensuring that user code and information remain protected. The architecture supports scalability and efficient handling of multiple tasks, making the system reliable for real-world usage.



V. Conclusion

In conclusion, the proposed Self-Hosted AI Coding Assistant successfully demonstrates a secure, efficient, and scalable solution for real-time code generation and development support. By integrating locally hosted AI models with a web-based development environment, the system eliminates dependency on external cloud services while ensuring data privacy and control. The platform combines essential features such as AI-driven code generation, interactive code editing, and secure execution within a unified interface, significantly enhancing the overall development experience.

The modular architecture of the system enables flexibility and future scalability, allowing easy integration of advanced features and improved AI models. The use of sandboxed environments ensures safe code execution, while the intuitive interface makes the platform accessible to both beginners and experienced developers. Overall, the system provides a reliable and privacy-focused alternative to traditional cloud-based coding assistants, contributing to increased productivity and secure software development practices.

VI. FUTURE SCOPE

1. **Advanced AI Model Integration:** Future enhancements can include the integration of more powerful and fine-tuned open-source models to improve code accuracy, context understanding, and multi-language support.
2. **Collaborative Development Environment:** The system can be extended to support real-time collaboration, allowing multiple users to work on the same project simultaneously, similar to modern cloud-based IDEs.
3. **Plugin and Extension Ecosystem:** Introducing a plugin-based architecture will enable developers to add custom tools, frameworks, and extensions, increasing the flexibility and usability of the platform.
4. **Voice-Based Coding Assistance:** Integration of voice recognition technology can allow developers to interact with the AI assistant using voice commands, improving accessibility and user experience.
5. **Enhanced Security Mechanisms:** Future improvements can include advanced security features such as code vulnerability detection, automated security audits, and stricter sandbox isolation techniques.
6. **Performance Optimization:** Further optimization of local model execution and resource management can enhance system performance, especially on low and mid-range hardware systems.
7. **Integration with Version Control Systems:** Incorporating version control features such as Git integration will allow users to manage code versions, collaborate efficiently, and track changes within the platform.

ACKNOWLEDGEMENT

The successful completion of this project would not have been possible without the guidance and support of several individuals. We would like to express our sincere gratitude to our project guide for providing valuable insights, continuous encouragement, and constructive feedback throughout the development of this work. Their expertise and mentorship played a crucial role in shaping the direction and quality of this project.

We also extend our heartfelt thanks to the faculty members and the department for their support, resources, and guidance during the course of this project. Their suggestions and technical knowledge greatly contributed to overcoming challenges encountered during development.

Finally, we would like to thank our peers and team members for their collaboration, dedication, and consistent efforts. Their contributions and teamwork were essential in successfully completing this project. We are also grateful to our families for their constant support and motivation throughout this journey.

REFERENCES

[1] T. Hua et al., "ResearchCodeBench: Benchmarking LLMs on Implementing Novel Machine Learning Research Code," *arXiv preprint arXiv:2506.02314*, 2025.

[2] H. Ma et al., "SRLCG: Self-Rectified Large-Scale Code Generation with Multidimensional Chain-of-Thought," *arXiv preprint arXiv:2504.00532*, 2025.

[3] E. Trofimova et al., "CodeRefine: A Pipeline for Enhancing LLM-Generated Code Implementations," *arXiv preprint arXiv:2408.13366*, 2024.

[4] K. Thangarajah et al., "Context-Aware CodeLLM Eviction for AI-Assisted Coding," *arXiv preprint arXiv:2506.18796*, 2025.

[5] "Usage of Large Language Models for Code Generation Tasks: A Review," *SN Computer Science (Springer)*, 2025.

[6] "LLM-Based Code Generation: A Systematic Literature Review," *IEEE Access*, 2024–2025.

[7] Z. Zhao et al., "Code Generation Using Self-Interactive Assistant," *IEEE COMPSAC*, 2024.

[8] C. Wang et al., "Context-Aware Multimodal AI Assistants," *ACM Multimedia*, 2023.

[9] J. Lin et al., "Integrating Visual-Language Models in Software Design," *IEEE Transactions on Multimedia*, 2024.

[10] R. Taylor et al., "Autonomous Multi-Agent Coding Frameworks," *ICLR Workshops*, 2024.

[11] L. Zhao et al., "Decentralized AI Pipelines for Privacy-Centric Automation," *IEEE Access*, 2024.

[12] P. Rana et al., "Explainability and Developer Trust in AI Tools," *CHI Conference on Human Factors in Computing Systems*, 2023.

[13] T. Verma et al., "Kubernetes-Based AI Orchestration for Scalable Systems," *IEEE Cloud Computing Conference*, 2023.

AUTHORS BIOGRAPHY

Hemant Vaidya,
Pursuing Final year in B.E. CSE (AI&ML) at Smt. Indira Gandhi College of Engineering, Ghansoli, New Mumbai, Maharashtra, India.

Kartik Nayak,
Pursuing Final year in B.E. CSE (AI&ML) at Smt. Indira Gandhi College of Engineering, Ghansoli, New Mumbai, Maharashtra, India.

Aditi Singh Thakur,
Pursuing Final year in B.E. CSE (AI&ML) at Smt. Indira Gandhi College of Engineering, Ghansoli, New Mumbai, Maharashtra, India.

Prof. Sarita Khedikar,
Professor of CSE-AIML, at Smt. Indira Gandhi College of Engineering, Ghansoli, Navi Mumbai, Maharashtra,India

Citation of this Article:

Hemant Vaidya, Kartik Nayak, Aditi Singh Thakur, Prof. Sarita Khedikar, "Self-Hosted AI Coding Assistant for Secure Real-Time Code Generation" Published in *International Journal of Scientific Research in Engineering and Management (IJSREM),* Volume 10, Issue 4, pp 87-91, APRIL 2026. Article DOI: 10.55041/IJSREM60018