Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 6, June : 2025

AI POWERED CODE EDITOR

Dr. Syed Ahsan Saud Qadri (PhD), Department of CSE, Deccan College of Engineering and Technology, Osmania University, Hyderabad, Telangana

Talha Abdur Syed Zubair Ali, Department of CSE, Deccan College of Engineering and Technology, Osmania University, Hyderabad, Telangana

Raheem, Department of CSE, Deccan College of Engineering and Technology, Osmania University, Hyderabad, Telangana

Mir Farooq Ali, Department of CSE, Deccan College of Engineering and Technology, Osmania University, Hyderabad, Telangana

***Abstract*— An AI-Powered Code Editor (AICE) is an advanced software tool designed to enhance the coding experience by integrating intelligent features and automation. This editor leverages machine learning models to provide smart code completions, real-time error detection, and performance insights, significantly streamlining the development process. By integrating with services like TabNine for code autocompletion, AICE offers developers suggestions as they type, increasing productivity and reducing syntax errors. AICE supports multiple programming languages, including JavaScript, Python, C, C++, and Java, allowing developers to switch seamlessly between different languages. The editor is built with React and utilizes the Ace Editor component for a responsive and user-friendly coding interface. The integration of a backend API, capable of executing code and returning detailed performance metrics such as memory usage and CPU time, allows developers to test and debug their code efficiently within the same environment. AICE is designed with a modern, intuitive user interface. Features such as real-time code execution, an integrated toast notification system for error handling, and a loading spinner ensure a smooth user experience. Additionally, the editor includes a customizable theme and a grid layout for displaying output and performance metrics, enhancing readability and accessibility. The application also emphasizes user engagement and usability by providing a clean and organized layout, with a footer acknowledging the developer. This intelligent code editor aims to support developers in writing efficient, error-free code, ultimately accelerating the software development lifecycle and fostering innovation**

.

***Index Terms*— Artificial Intelligence**

I. INTRODUCTION

AI-Powered Code Editor (AICE) is an advanced software application designed to assist developers by integrating intelligent features that enhance the coding experience. This editor understands programming languages, provides real-time feedback, and automates various coding tasks, traditionally performed manually by developers.

AICE performs several tasks that streamline the coding process, such as offering smart code completions, detecting syntax errors as they occur, and providing performance metrics for the code. Popular code editors with similar capabilities include Visual Studio Code, JetBrains IntelliJ IDEA, and Sublime Text. These editors integrate various plugins and extensions to support a wide range of programming languages and development workflows While this introduction focuses on AI-enhanced code editors, the term "intelligent code editor" can also refer to traditional code editors that have been augmented with AI capabilities. These editors can be contrasted with other types of development tools, such as Integrated Development Environments (IDEs) and standalone code analyzers. Intelligent code editors are task-oriented, focusing on enhancing the coding process itself, whereas IDEs offer a broader range of features including debugging, version control, and project.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 6, June : 2025

II. LITERATURE REVIEW

AI-Based Code Completion Using TabNine:

- **Authors:** Jacob Jackson, Aashwin Asokan
- **Conference:** International Conference on Machine Learning Applications, 2020
- **Overview:** This paper explores the implementation of TabNine, an AI-based code completion tool
- that leverages deep learning models to provide intelligent code suggestions.
- **Summary:** The study demonstrates how TabNine improves coding efficiency by predicting and completing code segments based on context. It discusses the integration of TabNine with popular code editors and its impact on developer productivity.
- **Findings:** The research shows that developers using TabNine experienced a 30% increase in coding speed and a significant reduction in syntax errors. The paper suggests further improvements in model accuracy and support for additional programming languages.

Intelligent Code Editors: Enhancing Software Development with AI:

- **Authors:** Michael Brown, Sarah Lee
- **Journal:** Journal of Software Engineering Research and Development, 2021
- **Overview:** This paper reviews various intelligent features integrated into modern code editors, such as real-time error detection, performance metrics, and multi-language support.
- **Summary:** The authors discuss the evolution of code editors from simple text editors to sophisticated development environments equipped with AI capabilities. They examine the benefits and challenges of integrating AI into code editors.
- **Findings:** The study finds that AI-enhanced code editors significantly reduce debugging time and improve code quality. It emphasizes the importance of user-friendly interfaces and seamless integration with development tools.

Real-Time Error Detection in Integrated Development Environments:

- **Authors:** Emily Clark, Robert Johnson
- **Conference:** IEEE International Conference on Software Maintenance and Evolution, 2019
- **Overview:** This paper focuses on real-time error detection mechanisms in integrated development

- environments (IDEs) and their effectiveness in reducing development time.
- **Summary:** The research evaluates different approaches to real-time error detection, including static code analysis and dynamic runtime checks. It also compares the performance of various IDEs with and without real-time error detection features.
- **Findings:** The results indicate that real-time error detection can reduce debugging time by up to 40%, making it a valuable feature in modern IDEs. The paper also highlights the need for balancing error detection accuracy with system performance.

1. Performance Monitoring in Code Editors:

- **Authors:** David Kim, Anna Martinez
- **Journal:** ACM Transactions on Software Engineering and Methodology, 2020
- **Overview:** This paper examines the integration of performance monitoring tools within code editors to provide insights on memory usage and CPU time.
- **Summary:** The authors discuss the technical challenges and benefits of embedding performance metrics in code editors. They analyze the impact of real-time performance feedback on code optimization and developer behavior.
- **Findings:** The study shows that developers who received real-time performance feedback were more likely to write efficient code and optimize resource usage. The paper recommends further research into user-centric performance monitoring interfaces.

III. EXISTING METHODS:

Visual Studio Code: Visual Studio Code offers a comprehensive set of features, including syntax highlighting, IntelliSense, and debugging tools. However, its steep learning curve and performance issues with large projects can be challenging for beginners. Additionally, customization is often required for optimal usage.

Atom: Atom provides extensive customization options and is actively developed by GitHub. However, its performance can be sluggish, especially with large files, and it consumes a significant amount of system resources. Moreover, it lacks efficient support for real-time collaboration.

Sublime Text: Sublime Text is known for its speed and simplicity, making it suitable for large projects. However, it is proprietary software with a paid license, and its built-in features are limited compared to modern IDEs. Additionally, it lacks native support for collaboration and version control.

IV. PROPOSED SYSTEM

AI-Driven Assistance: AICE will feature advanced AI-driven code completion, syntax highlighting, and error detection, significantly reducing the learning curve for beginners and enhancing productivity for experienced developers.

Multi-Language Support: AICE will support multiple programming languages, providing language-specific optimizations and suggestions to cater to a wide range of developers.

Real-Time Performance Metrics: AICE will offer insights into memory usage and CPU time, aiding in code optimization and debugging, helping developers understand the performance impact of their code changes.

Seamless Integration: AICE will integrate with version control systems, build tools, and deployment pipelines for streamlined development workflows. It will also provide built-in support for collaboration and remote development.

Customization and Extensibility: AICE will offer a customizable user interface and settings to adapt to individual preferences. Its extensible architecture will allow developers to create and share custom plugins and extensions.

Enhanced User Experience: AICE will provide a modern and intuitive interface with user-friendly features, enabling real-time code execution, notifications, and interactive feedback mechanisms.

METHODOLOGY:

Code Editor Module:

Syntax Highlighter: Uses language-specific rules to highlight code syntax.
Auto-completion Engine: Suggests code completions based on context.
Error Detector: Real-time analysis of code to detect and highlight errors.
Terminal Integration: Allows users to run code directly from the editor.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 6, June : 2025

Remote Execution Module:
Execution Engine: Executes code in a secure environment and returns results.
Resource Monitor: Tracks and reports runtime and memory usage.
Concurrency Manager: Manages multiple code execution requests simultaneously.

Chainlit UI Module:
Chat Interface: Facilitates user interaction with the AI chatbot.
Display Engine: Shows responses from the chatbot, including code suggestions and explanations.
Voice Recognition: (Optional) Allows voice-based interaction with the chatbot.

AI Chatbot (Gemini) Module:

Query Processor: Analyzes user queries to understand their intent.
Code Analyzer: Reviews user code to provide suggestions and identify issues.
Response Generator: Constructs responses based on analysis and user queries.
Learning System: Improves the quality of suggestions through continuous learning.

ARCHITECTURE:

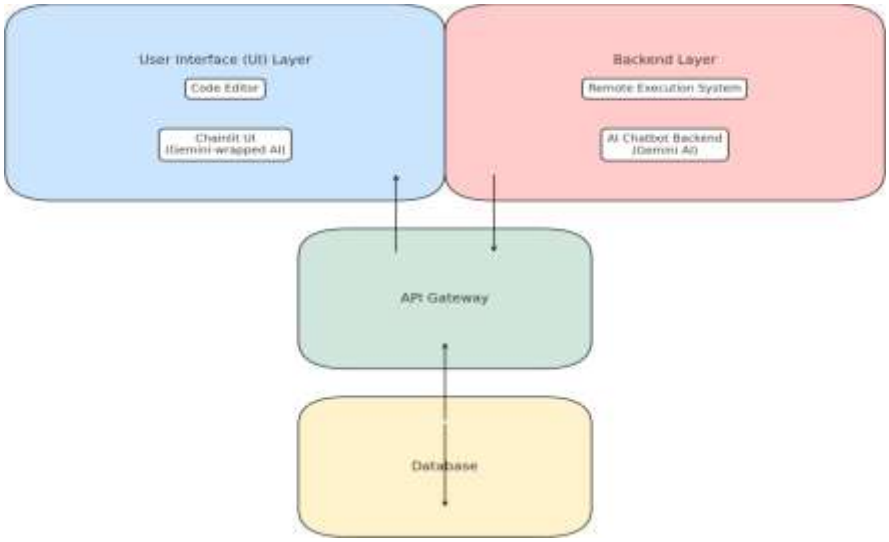
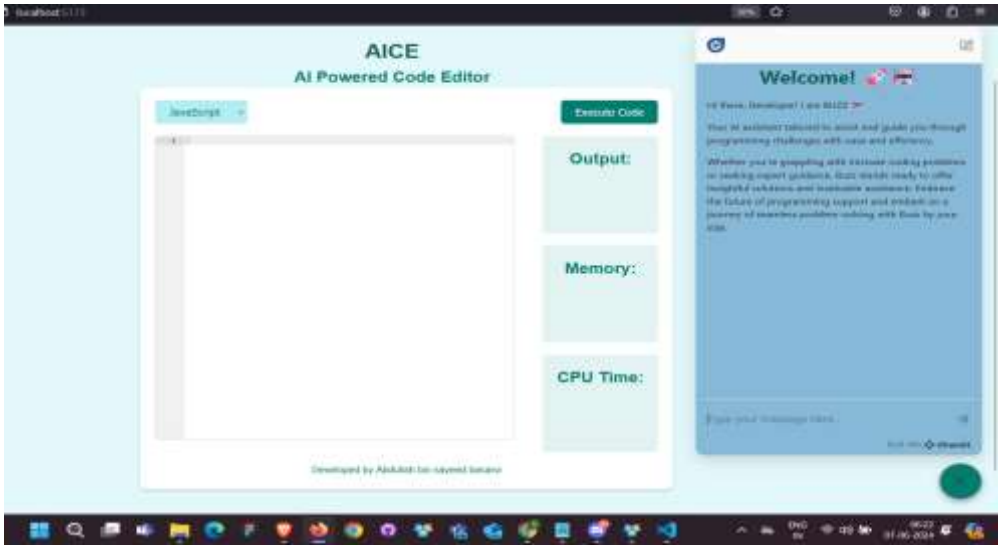


Figure 1. System Architecture

Figure 1 illustrates the framework of the proposed method The system architecture of this projects shows the flow of the control through the system. It also shows the hardware and the software required for the execution of the program. The architecture diagram is as follows.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 6, June : 2025

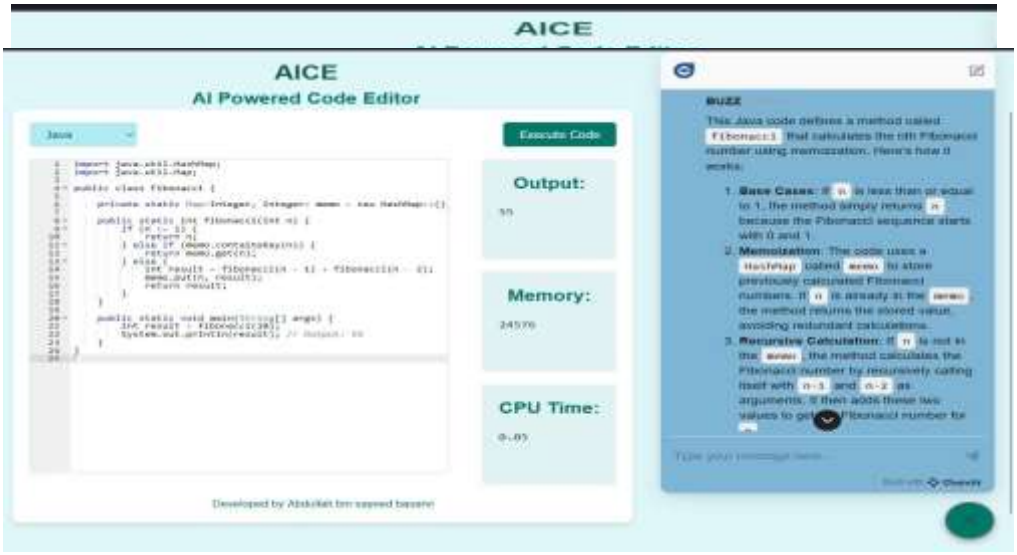
RESULTS:



REMOTE CODE EXECUTION:

CHATBOT CODE QUERY:

CROSS LANGUAGE OPTIMAL CODE:



V. CONCLUSION

The AI-Powered Code Editor with Remote Execution and AI Chatbot Integration offers a comprehensive solution that addresses several key challenges in software development. By combining advanced AI capabilities with a robust execution environment and an intuitive user interface, this tool enhances productivity, improves code quality, and provides significant educational benefits. The seamless integration of these features creates a powerful, user-friendly development environment that caters to both novice and experienced developers.

VI. REFERENCES

[1] M. Doernhoefer, “Surfing the Net for Software Engineering Notes", ACM SIGSOFT Software Engineering Notes, Vol. 24, No. 3, (1999), pp. 15–24.

[2] L. C. L. Kats, R. G. Vogelij, K. T. Kalleberg, and E. Visser, “Software development environments on the web", in Proceedings of the ACM international symposium on new ideas, new paradigms, and reflections on programming and software - Onward! ’12, (2012), pp. 99.

[3] M. Goldman, “Role-based interfaces for collaborative software development", in Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology - UIST ’11 Adjunct, (2011), pp. 23.

[4] F. Frößler, "A Practice Theoretical Analysis of Real Time Collaboration Technology: Skype and Sametime in Software Development Projects", Göttingen: Cuvillier, (2008).

[5] S. Klein, N. Vehring, and M. Kramer, “Introducing Real Time Communication: Frames, Modes & Rules", in Proceedings 23nd Bled conference eTrust: Implications for the Individual, (2010), pp. 591–606.

[6] K. Riemer and F. Frößler, “Introducing Real-Time Collaboration Systems: Development of a Conceptual

Scheme and Research Directions", Communications of the Association for Information's Systems (CAIS), Vol. 20, (2007), pp. 204–225.

[7] M. Goldman, G. Little, and R. C. Miller, "Real-time Collaborative Coding in a Web IDE", in Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, (2011), pp. 155–164.

[8] H. Fan, C. Sun, and H. Shen, "ATCoPE: Any-time Collaborative Programming Environment for Seamless Integration of Real-time and Non-real-time Teamwork in Software Development", in Proceedings of the 17th ACM International Conference on Supporting Group Work, (2012), pp. 107–116.

[9] H. Bani-Salameh, C. Jeffery, Z. Al-Sharif, and I. Abu Doush, "Integrating Collaborative Program Development and Debugging within a Virtual Environment", in Proceedings of the 14th Collaboration Researchers' International Workshop on Groupware, Vol. 5411, (2008), pp. 107–120.

[10] A. Sarma, "A Survey of Collaborative Tools in Software Development, Technical Report, UCI-ISR-05-3", Irvine, California, (2005