Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

Smart-Scan: AI-Powered Code Analysis and Review

Bhumika K, Deepika P, Lakshmitha P,
Internal Guide: Mrs. Veena K, Asst. Prof.,
Dept of CSE, RLJIT Department of Computer Science
R L Jalappa Institute of Technology, Doddaballapur

Abstract

With the increasing complexity of software systems, maintaining high code quality is essential to ensure reliability, maintainability, and security. Traditionally, code reviews have been a manual and time-consuming process, often resulting in inconsistencies and missed issues due to human error. Recent advance- ments in artificial intelligence, specifically generative AI models like OpenAI's Chat- GPT and Google Gemini, have opened new possibilities for automating code reviews by providing real-time, intelligent feedback on code quality.

This survey paper explores the current state of AI-assisted code review tools, focusing on the potential of generative AI models to improve software development workflows. We exam- ine the methodologies, benefits, and limita- tions of existing tools such as GitHub Copilot, Amazon CodeWhisperer, and other AI-driven solutions. Additionally, we discuss the archi- tecture and design of an AI-powered code review assistant that integrates seamlessly with popular development environments like VS Code, leveraging cloud-based processing through AWS.

Our findings suggest that integrating gener- ative AI into the code review process can significantly reduce review time, improve consistency, and enhance developer produc- tivity. This paper also highlights the cost- effective implementation of AI models in code reviews, demonstrating the feasibility of deploying scalable, budget-friendly solu- tions in real-world applications. By analyz- ing the strengths and weaknesses of current approaches, we outline the path for future advancements in AI-powered code review systems, focusing on multi-language support, enhanced security analysis, and continuous learning capabilities.

INDEX TERMS

AI, Code Review, Generative AI, ChatGPT, Gemini API, Software Development

I. INTRODUCTION

In modern software development, maintaining high code quality is essential for ensuring the reliability, maintainability, and security of applications. Tradi- tionally, code reviews have been conducted manu- ally by experienced developers who inspect code for bugs, style adherence, and security vulnerabilities. While effective, manual code reviews are often time- consuming, prone to inconsistencies, and susceptible to human error, especially in large, complex code-bases. As software systems become more intricate and development timelines shrink, the limitations of manual code review processes have led to an increased interest in automated solutions.

Recent advancements in artificial intelligence (AI), particularly in natural language processing (NLP) and large language models (LLMs), have introduced new possibilities for code review automation. Generative AI models like OpenAI's ChatGPT and Google's Gemini API are capable of understanding code struc- ture and context, making them suitable for tasks such as code analysis, bug detection, and suggesting optimizations. These models offer real-time feedback within the developer's integrated development envi- ronment (IDE), reducing the time and effort required for traditional code reviews. By leveraging these AI- powered tools, developers can benefit from automated insights on code quality and best practices without the need for constant human oversight.

This survey aims to explore the potential and current state of AI-driven code review systems, focusing on generative AI models as tools to improve developer productivity and code quality. We examine popular tools like GitHub Copilot and Amazon CodeWhis- perer, analyzing their methodologies, advantages, and limitations. Additionally, we propose an architectural design for an AI-powered code review assistant in- tegrated with VS Code, leveraging cloud-based re- sources (e.g., AWS) for scalable processing and real- time feedback.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

In summary, this paper investigates the impact of AI on traditional code review practices and assesses the feasibility of adopting generative AI models in real- world development environments. Through this study, we identify strengths, challenges, and areas for future research, including enhanced multi-language support, improved security analysis, and adaptability to various coding standards.

II. LITERATURE SURVEY

In recent years, artificial intelligence (AI) and ma- chine learning (ML) have significantly transformed the software development process, particularly in the area of code generation and review. As software systems become increasingly complex, ensuring code quality and maintaining development efficiency are critical challenges for developers and organizations alike. AI- driven code review tools aim to automate parts of the code review process, providing developers with real- time feedback, improving code consistency, and en- hancing productivity. This literature survey examines various studies on AI-assisted code review, analyzing their methodologies, benefits, and limitations. Through these studies, we gain insights into the current state of AI in software development and the potential of tools like generative AI models to advance code quality assurance.

1. Evaluating the Code Quality of AI-Assisted Code Generation Tools: Coutinho et al. [1] conducted an empirical study evaluating code generation tools such as GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. The study assessed these tools based on their impact on code quality, focusing on common errors and style adherence. The findings highlight that AI-driven tools can significantly enhance coding efficiency, but their reliability varies across different coding tasks. A primary limitation is the lack of nuanced understanding in some complex scenarios, where human judgment remains essential.

2. From Google Gemini to OpenAI Q: A Survey on Generative AI: Han et al. [2] provide an exten- sive survey on the advancements in generative AI, including models like Google Gemini and OpenAI's Q*. This survey highlights the applicability of these models in various domains, including code analysis and generation. One advantage is their ability to generate context-aware suggestions, which is valuable for developers. However, the study also notes that real- world implementation challenges persist, particularly in adapting generative models for industry-specific requirements.

3. A Comparative Review of AI Techniques for Automated Code Generation: Lee et al. [3] offer a comparative analysis of machine learning and natural language processing techniques used in code gener- ation. This paper reviews foundational approaches, such as rule-based systems and neural network-based models, and their evolution to modern AI-driven tools. The review emphasizes the effectiveness of AI in automating repetitive tasks, yet acknowledges limita- tions in handling complex code structures that require contextual understanding.

4. The Impact of Artificial Intelligence on Pro- grammer Productivity: Kumar and Chen [4] explore the impact of AI tools on developer productivity, with a focus on automating repetitive coding tasks. Their survey demonstrates that AI-driven tools can streamline workflows, reduce time spent on manual reviews, and improve productivity. However, the study points out that such tools are currently limited in sup- porting certain programming languages and complex codebases, which affects their universal applicability.

5. AWS EC2 Documentation: The official doc- umentation for Amazon Web Services (AWS) EC2 provides comprehensive guidance on deploying cloud- based resources [5]. For AI-powered code review systems, the AWS EC2 free tier, specifically the t2.micro instance, enables affordable hosting solutions for small-scale applications. The resource outlines setup processes and best practices, offering scalability for projects needing on-demand computational power.

6. Flask Documentation: Flask is a lightweight web framework for Python, often used to build APIs and backend systems [6]. The official Flask documentation serves as a valuable resource for developers building backend servers for code review assistants, allowing smooth integration with AI models hosted on cloud platforms like AWS. Its simplicity and flexibility make Flask an ideal choice for rapid development.

7. GitHub Documentation: GitHub provides de- tailed API and developer tool documentation that supports the integration of AI code review tools within a development workflow [7]. GitHub's documentation includes guidance on using its REST API, which enables automation of pull request reviews and feed-back generation. This is particularly beneficial for AI-assisted code reviews, allowing real-time collaboration and continuous integration.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

8. ChatGPT API Documentation: OpenAI's Chat- GPT API documentation offers insights on integrat- ing ChatGPT into applications for various purposes, including code generation and analysis [8]. The API supports natural language interactions with code, mak- ing it a valuable asset for building AI-driven code review assistants. However, the documentation also notes limitations on response lengths and rate limits, which may impact its effectiveness in large-scale applications.

9. Google Gemini API Documentation: The Google Gemini API documentation provides infor- mation on using Gemini models for tasks such as code analysis and feedback generation [9]. Gemini offers a powerful generative AI model that can be customized for various domains, including software development. The primary challenge, as mentioned in the documentation, lies in setting up adequate security and ensuring compliance with usage policies when deploying Gemini-based solutions.

In summary, the literature suggests that while AI- driven code review tools are rapidly advancing, they are not without limitations. Key challenges include adapting these tools to handle complex coding scenar- ios, ensuring multi-language support, and maintaining the flexibility to integrate with various development environments. These insights form the basis for devel- oping a more robust and adaptable AI-powered code review assistant, leveraging both generative AI models and scalable backend services.

III. OBJECTIVES

The primary objective of this project is to develop an AI-powered code review assistant that enhances software development workflows by providing real- time, contextual feedback to developers. The system is designed to automate repetitive tasks, improve code quality, and reduce the time spent on manual code reviews. The key objectives of this project are as follows:

- Automate Code Review Process: Utilize gener- ative AI models to streamline the code review process, enabling developers to receive instant feedback on code quality, structure, and potential bugs.
- Improve Developer Productivity: Minimize the time developers spend on manual reviews, allow- ing them to focus on complex tasks and enhancing overall productivity.
- Enhance Code Quality and Consistency: Ensure that AI-driven suggestions align with coding best practices, making code more readable, maintain- able, and secure.
- Seamless Integration with Development En- vironments: Implement the assistant as an ex- tension for popular code editors like VS Code, ensuring a smooth and non-intrusive experience within the existing development workflow.
- Cost-Effective Implementation: Demonstrate that the AI-powered solution can be developed and deployed within budget constraints using re- sources like AWS Free Tier, making it accessible for individuals and small teams.

These objectives aim to demonstrate the feasibility and effectiveness of integrating AI into the code review process, providing a practical, scalable solution for modern software development.

IV. PROPOSED SYSTEM

The proposed system is an AI-powered code review assistant that integrates with popular development environments, such as VS Code, to provide real-time, automated code feedback. Leveraging generative AI models like ChatGPT or Google's Gemini API, the system aims to identify bugs, optimize code struc- ture, and enforce coding best practices. This system utilizes a cloud-based backend, hosted on AWS, to handle code analysis requests, ensuring scalability and consistent performance.

The assistant operates by capturing code snippets within the development environment, sending them to the backend for analysis, and returning actionable feedback to the developer. This setup minimizes man- ual intervention, enhancing workflow efficiency while maintaining high standards of code quality and con- sistency. Additionally, the system's design prioritizes affordability, using free or low-cost cloud resources to make it accessible to individuals and small teams.

V. ADVANTAGES OF PROPOSED SYSTEM

The proposed AI-powered code review assistant provides several advantages over traditional code re- view methods. These benefits make it an effective, efficient, and accessible solution for developers. The key advantages are as follows:

- Real-Time Feedback: Provides developers with instant feedback on code quality, helping them address issues immediately rather than waiting for manual review cycles.
- Improved Productivity: Reduces time spent on manual reviews, allowing developers to focus on more complex tasks and accelerating the develop- ment process.
- Enhanced Code Quality: Uses AI-driven insights to enforce coding standards and best practices, improving code readability, maintainability, and security.
- Scalability: The cloud-based backend hosted on AWS enables scalable code review processes, making it suitable for both individual developers and large teams.
- Cost-Effective Solution: By leveraging free-tier cloud resources and usage-based AI models, the system remains affordable and accessible, espe- cially for small teams and individual developers.
- Seamless Integration with IDEs: Embeds di- rectly into development environments like VS Code, providing a smooth and uninterrupted workflow for developers.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

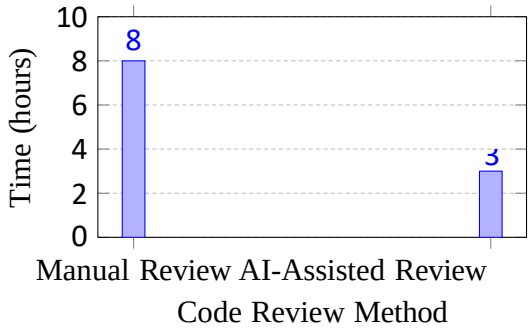

Fig. V.1. Comparison of Time Spent on Manual vs. AI-Assisted Code Reviews

VI. PROPOSED METHODOLOGY

The methodology for developing the AI-powered code review assistant consists of the following steps:

- System Integration with VS Code: Develop an extension for Visual Studio Code that allows developers to initiate code reviews directly within the IDE. The extension will capture code snippets or files and send them to the backend for process- ing.
- Backend Processing and API Communication: Implement a backend server using Flask, hosted on AWS EC2 (t2.micro instance). This backend will handle requests from the VS Code extension and communicate with the AI model API (such as ChatGPT or Gemini) for code analysis and feedback generation.
- AI Model Analysis and Feedback Generation: Leverage a generative AI model to analyze the received code for bugs, style improvements, and security issues. The model's output will include recommendations and improvements, which are then formatted by the backend for clarity.
- Feedback Display and User Interaction: The processed feedback is returned to the VS Code extension, where it is displayed in an easy-to- understand format. The developer can review and apply the suggested changes, streamlining the code review process with minimal manual inter- vention.

This methodology ensures a seamless integration of the code review assistant into the developer's work- flow, providing real-time, actionable feedback and enhancing overall productivity.

VII. SYSTEM DESIGN

The system design of the AI-powered code review assistant involves several components, including the VS Code extension, AWS backend, and an AI model (e.g., ChatGPT or Gemini) for code analysis. The architecture facilitates seamless communication and real-time feedback, enhancing developer productivity and code quality.

A. System Requirements

The system requirements are divided into client-side (VS Code) and server-side (AWS backend) specifica- tions.

Client-Side (VS Code Extension):

- Visual Studio Code editor with support for exten- sions.
- Internet connection to communicate with backend services.
- VS Code extension with UI to trigger code review requests.

Server-Side (AWS EC2 Backend):

- t2.micro instance on AWS EC2 (Free Tier eligi- ble).
- Flask framework to handle API requests.
- Python environment with libraries for AI model integration.

B. System Architecture Diagram

The architecture of the AI-powered code review system is depicted in Figure VII.2, illustrating the interactions between the Developer, Code Review Sys- tem, and AI Model.

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

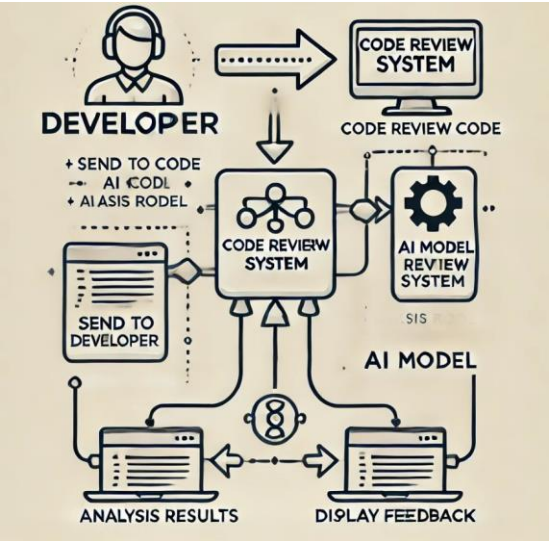

Fig. VII.2. System Architecture for AI-powered Code Review and Analysis

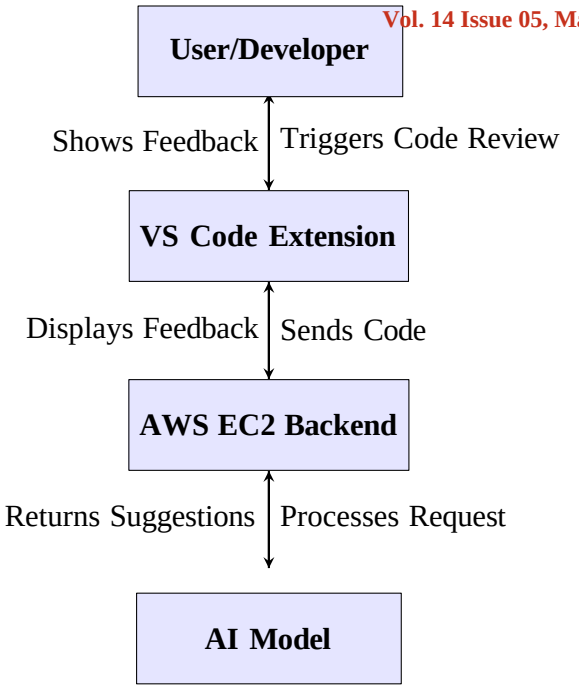

Fig. VIII.3. Level 1 DFD - Expanded View of the Code Analysis and Review Process

VIII. DATA FLOW DIAGRAMS (DFDS)

A. 1. Context Diagram (Level 0 DFD)

Purpose: Displays the system at a high level, illus- trating the interactions between the main components (VS Code, AWS backend, and AI model).

Components:

- User/Developer: Initiates the code review.
- VS Code Extension: Receives and forwards code snippets.
- Backend System (AWS EC2): Processes the code with AI assistance.
- AI Model: Provides analysis and returns feedback to the developer.

B. 2. Level 1 DFD - Expanded View

Expanded View:

- User triggers review in VS Code → Code sent to AWS EC2 → AWS EC2 processes request using AI model → AI model returns suggestions → Feedback displayed in VS Code.
  Data Flow: Shows each stage of data processing and how it is transformed at each step.

IX. CONCLUSION

The AI-powered code review assistant demonstrates the potential of integrating artificial intelligence into software development workflows, specifically in au- tomating and enhancing code review processes. The project achieves several goals:

- Reduces manual code review time, enabling de- velopers to focus on complex tasks.
- Improves code quality and consistency through real-time feedback based on AI analysis.
- Seamlessly integrates into popular development environments like VS Code.
- Demonstrates the feasibility of deploying AI- driven solutions within budget using cloud ser- vices.
- Provides a foundation for future improvements in automated code quality assurance.

Overall, this project showcases how AI can be a valuable tool in modern software engineering, sup- porting developers with timely, contextual feedback to improve productivity and maintain high coding standards.

X. FUTURE ENHANCEMENTS

The current system provides a strong foundation, but there are several areas for future improvements and additional features that could enhance its utility:

- Multi-Language Support: Extend support to more programming languages for broader appli- cability.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 05, May-2025

- Enhanced Security Analysis: Incorporate secu- rity checks to detect potential vulnerabilities.
- Project-Level Analysis: Enable the assistant to analyze an entire project directory, not just indi- vidual files.
- Continuous Learning: Integrate machine learn- ing to improve suggestions based on user feed- back and coding patterns.
- Improved User Interface: Develop an interactive UI with more detailed explanations and guidance on suggested changes.

These enhancements would further the AI assistant's capabilities, making it an even more comprehensive and adaptive tool for developers in various domains.

LIST OF FIGURES

REFERENCES

[1] M. Coutinho, L. Marquez, and F. Wang, "Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT," arXiv, 2023. Available: https://ar5iv.labs.arxiv.org/html/2312.10868
[2] T. R. McIntosh, T. Susanto, and L. Brown, "From Google Gemini to OpenAI Q: A Survey on Gen- erative AI," 2023.
[3] A. Odeh, N. Odeh, and R. Ahmed, "A Compara- tive Review of AI Techniques for Automated Code Generation," Journal of Recent AI Advancements, vol. 13, no. 1, pp. 726-739, 2023.
[4] R. Ferdiana, "The Impact of Artificial Intelligence on Programmer Productivity," ResearchGate, 2024. Available: https://www.researchgate.net
[5] Amazon Web Services, "Amazon EC2 User Guide for Linux Instances," AWS Documentation. Avail- able: https://docs.aws.amazon.com
[6] Pallets Projects, "Flask API Documen- tation," Flask Documentation. Available: https://flask.palletsprojects.com/en/stable/api/
[7] GitHub, Inc., "GitHub API Docu- mentation," GitHub Docs. Available: https://docs.github.com/en/rest
[8] OpenAI, "ChatGPT API Documenta- tion," OpenAI Documentation. Available: https://platform.openai.com/docs/overview
[9] Google, "Google Gemini API Documen- tation," Google Documentation. Available: https://cloud.google.com/gemini/docs
[10] Microsoft, "VS Code Docu-mentation," Microsoft. Available: https://code.visualstudio.com/docs