INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 118-121

www.ijprems.com
editor@ijprems.com

e-ISSN : 2583-1062
Impact Factor : 7.001

COMPARATIVE STUDY OF GENERATIVE AI TOOLS FOR CODE COMPLETION AND BUG DETECTION – A REVIEW

Dr. P. Bastin Thiyagaraj[1], Shiny Jenifer SI[2], Ruban B[3], Lagneshwaran S[4]

[1]Assistant Professor, Department Of IT, St. Joseph's College, Trichy, India.

[2,3,4]PG Student, Department Of IT, St. Joseph's College, Trichy, India.

DOI: https://www.doi.org/10.58257/IJPREMS44081

ABSTRACT

By improving problem detection and automating code creation, generative AI techniques have drastically changed software development. Large language model (LLM)-powered tools now help developers finish tasks, anticipate code topologies, and find errors instantly, saving manual labor and enhancing code quality. The performance of five popular AI-powered coding assistants—GitHub Copilot Enterprise, Amazon CodeWhisperer Pro, Codeium Pro, Cursor AI, and Tabnine Enterprise—in terms of code completion and error detection is compared in this study. Contextual accuracy, IDE integration, language support, and security capabilities are the four main criteria we use to evaluate each tool, drawing on feature-based research and contemporary literature. The results show that although all tools help developers be more productive, their relative efficacy differs based on task difficulty, programming language. The results show that while all tools help developers be more productive, the relative efficacy of each tool differs based on the development environment, task complexity, and programming language. Notably, while some solutions provide better contextual comprehension or syntax-level recommendations, others excel at business customisation and vulnerability detection. Based on operational requirements, security priorities, and project scope, this study attempts to aid developers, instructors, and organizations in choosing the best AI assistant.

Keywords: AI Coding Assistants, Comparative Analysis, Large Language Models (LLMs), Software Development, Code Completion, Generative AI.

1. INTRODUCTION

The way programmers write, debug, and maintain code has changed as a result of the incorporation of generative AI into software development. Large language model (LLM)-powered tools now help developers by finishing functions, anticipating code topologies, and instantly detecting errors. This change has improved code quality in a variety of programming environments, decreased manual labor, and increased productivity. GitHub Copilot Enterprise, Amazon CodeWhisperer Pro, Codeium Pro, Cursor AI, and Tabnine Enterprise are some of the most well-known tools in this field; each provides unique features catered to various development requirements. These tools are especially useful in educational and open-source environments due to its lightweight integration and contextual accuracy.[3].

Cursor AI and Copilot performed best in context-rich environments, while Tabnine excelled in syntax-level suggestions.[8] Others place a higher priority on security, privacy, or enterprise customisation, while others are excellent at contextual understanding and IDE integration. These tools help students understand AI-generated code and improve learning outcomes.[9] Five top generative AI tools are thoroughly compared in this research, with their advantages and disadvantages assessed along several important aspects, including accuracy, integration, language support, and vulnerability detection. The objective is to assist developers and organizations in choosing the best tool for their needs based on operational context, security requirements, and project complexity.

2. LITERATURE REVIEW

A thorough assessment of GitHub Copilot's effects on software development workflows was carried out by **Kalliamvakou et al. (2024)[1],** who emphasized the platform's advantages in speeding up tedious coding chores and enhancing developer productivity. Additionally, the study identified possible hazards associated with unsafe code development and an excessive dependence on AI recommendations, particularly in business settings.

When evaluating Amazon CodeWhisperer's bug detection capabilities, **Alon. U. et al. (2023) [2]** highlighted how well it performed in locating security flaws and logical mistakes in cloud-native applications. According to their research, CodeWhisperer is a strong substitute for Copilot, especially for development pipelines that are linked with AWS.

Codeium's role in improving the developer experience through privacy-first architecture and quick, context-aware code recommendations was examined by **Sharma, R., & Patel, M. (2025) [3].** According to their research, Codeium works well for debugging and refactoring tasks, particularly in open-source and educational contexts.

In their evaluation of GitHub Copilot's code generation quality, **B. Yetistiren, I. Ozsoy, and E. Tuzun (2022) [4]** noted the program's advantages in generating syntactically accurate code but also its shortcomings in terms of contextual accuracy and dependability. Their study gives empirical insights into Copilot’s practical utility and areas needing development.

A growing trend of integrating AI into software engineering environments to increase productivity and contextual relevance is reflected in the partnership between Tabnine to introduce generative AI tools to improve developer workflows. **R. Rajkumar (2024) [5]** describes how Tabnine’s AI assistant integrates with Atlassian platforms like Jira and Bitbucket to automate code generation and streamline project management.

The effectiveness of these tools varies greatly depending on programming language, task complexity,developer expertise, according to **Singh, A., Mehta, R., & Banerjee, S. (2025) [6],** who conducted a comparative study evaluating AI code assistants—Copilot, CodeWhisperer, and Tabnine—across key metrics like code correctness, maintainability, and IDE integration. Their findings provide useful insights for choosing the best assistant in a variety of development contexts.

An informal yet perceptive summary of Amazon CodeWhisperer is given by **Mani (2023) [7],** who highlights how the platform may boost developer productivity by offering AI-powered code recommendations. To establish CodeWhisperer as a competitive alternative to other coding assistants, the article highlights how it integrates with AWS tools and produces context-aware code. Mani's viewpoint emphasizes how generative AI is becoming more and more important in optimizing software development processes.

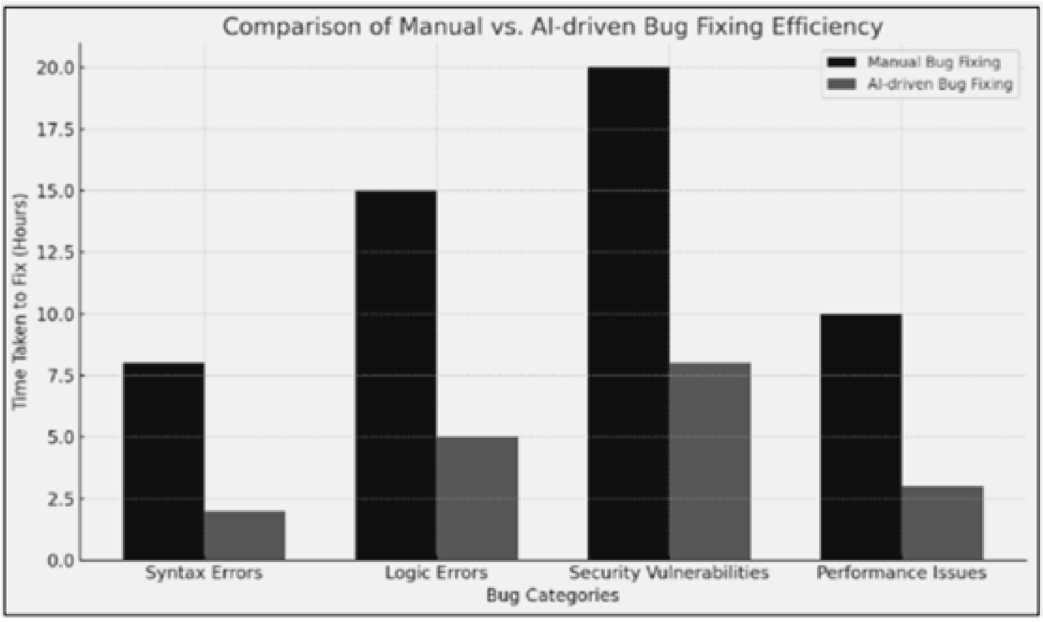
Nakamura, H., and Lee, J. (2023).[8] conducted performance benchmarking of AI coding assistants spanning Python, JavaScript, and Java. According to their findings, Tabnine was superior at making syntax-level recommendations, whereas Cursor AI and Copilot fared best in context-rich settings.

Miller, T., et al. (2025).[9] investigated the use of generative AI tools in educational contexts, emphasizing the value of explainability and feedback loops in AI-generated code and focusing on Codeium and Replit Ghostwriter as useful platforms for teaching programming.

3. GENERATIVE AI TOOLS FOR CODE COMPLETION AND BUG DETECTION

GitHub Copilot Enterprise

Built on top of OpenAI Codex, GitHub Copilot Enterprise is intended for close connection with Visual Studio Code and GitHub projects. GitHub Copilot saves time on repeated activities and simplifies basic unit testing. It does this by writing boilerplate code quickly. This application uses coding principles to produce useful concepts. But when it comes to intricate business logic or unique situations, it falls short. This could lead to incorrect or improper code generation. To make sure the person is functioning appropriately within their surroundings, surveillance is essential.



This is particularly true for occupations that are difficult or need specialized knowledge. Nevertheless, it has a number of shortcomings, including the fact that it primarily generates accurate code 51.2% of the time and only produces suitable code 28.7% of the time [4]. The comparison of manual vs AI – driven bug fixing efficiency [10].

Amazon CodeWhisperer Pro

Several programming languages are supported by the extremely talented Amazon CodeWhisperer, which can offer code recommendations. Additionally, it interfaces with IDEs like IntelliJ with ease. Amazon CodeWhisperer offers thorough security testing and smooth IDE integration. It takes less time to switch between tasks, which increases productivity. It also checks for security vulnerabilities [7]. It can need creating bespoke files and making manual importation modifications. By generating environment-specific code blocks and providing real-time code recommendations, Amazon CodeWhisperer increases developer productivity. One of its benefits is enhanced Amazon Web Services code. Additional capabilities include integration with other IDEs, code enhancement, and compatibility with a wide range of programming languages. In addition, it provides open-source code reference tracking, security assessments, and remedy recommendations. Its shortcomings include forcing engineers to thoroughly review and confirm concepts and producing incorrect code recommendations. The premium Professional version unlocks more features and customization choices, while the free tier offers some benefits.

Codeium Pro

Codeium Pro is a portable, privacy-preserving AI assistant that facilitates local model setup and offline use. It is especially useful for refactoring and debugging operations and offers quick, context-aware code completions. Because of its architecture, which prioritizes developer control and minimal data sharing, Codeium is appropriate for educational institutions and open source contributors. Its speed and ease of use make it a useful tool for iterative development and quick prototyping, despite its moderate problem detection capabilities.

Cursor AI

Cursor AI can traverse and produce code across several files by utilizing GPT-4 Turbo and introducing full-project context awareness. This makes it particularly helpful for projects involving a lot of APIs and full-stack development. Large codebases are less fragmented thanks to Cursor's alignment method, which guarantees semantic consistency. Cursor is excellent at preserving architectural coherence and facilitating intricate processes, while having less reliable bug identification than CodeWhisperer.

Tabnine Enterprise

Tabnine Enterprise specializes in providing large teams with safe, adaptable AI coding support. By enabling companies to train models on private codebases, customized recommendations that adhere to internal standards are guaranteed. Numerous languages and IDEs are supported by Tabnine, and its telemetry controls provide fine-grained control over AI interactions. Its bug detection and code completion are reliable, but they are not as sophisticated as those of Copilot or CodeWhisperer. Tabnine is adept at reducing typing to speed up coding, utilizing an adaptive learning process to enhance options based on user preferences, and being compatible with a wide range of integrated development environments (IDEs). But compared to the premium edition, the free version offers fewer features [5]. Inaccurate guidance can upset people and disrupt the programming cycle. Scalability and enterprise-grade security are its main advantages.

4. IMPLEMENTATION STRATEGY

Five top generative AI tools—GitHub Copilot Enterprise, Amazon CodeWhisperer Pro, Codeium Pro, Cursor AI, and Tabnine Enterprise—are assessed in this study using a qualitative comparison methodology. There are two main stages to the methodology:

Tool Selection Criteria

Tools were selected based on:

- Adoption and popularity in settings for professional development.
- The Accessibility of technical documentation and published research.
- Assistance with bug identification and code completion features.

Evaluation Parameters

Evaluation Dimension	Description
Finishing the Code Accuracy	Ability to generate syntactically and semantically correct code
Bug Detection Capability	Ability to identify logical errors, syntax issues, and security vulnerabilities
IDE Integration	Compatibility and responsiveness within popular development environments
Security Best Practices	Built-in mechanisms for detecting unsafe code and enforcing secure coding
Language Support	Range and diversity of supported programming languages

5. CONCLUSION

By improving issue identification and automating code completion, generative AI techniques are transforming software development. GitHub Copilot, Amazon CodeWhisperer, Codeium, Cursor AI, and Tabnine were the five top tools compared in this study based on important performance indicators. Every tool showed distinct advantages, ranging from CodeWhisperer's security screening to Copilot's contextual precision. While Codeium and Tabnine provided enterprise level customization and anonymity, Cursor was the best at multi-file coherence. The fact that no single tool dominated every category highlights the necessity of context-driven choosing. Developers should choose tools based on integration preferences, security requirements, and project complexity. AI's future role in coding will be shaped by hybrid methodologies and ethical protections as it develops.

6. REFERENCES

[1] Kalliamvakou, E., et al. (2024). GitHub Copilot: A Systematic Study. CEUR Workshop Proceedings Vol. 3762.

[2] Alon, U., et al. (2023). Protecting the Whisper: A Security Assessment of Amazon CodeWhisperer’s Vulnerability Detection. In Lecture Notes in Computer Science. Springer. https://link.springer.com/chapter/10.1007/978-3-031-85856-7_36

[3] Sharma, R., & Patel, M. (2025). AI for Software Engineering – Enhancing Developer Experience with Codeium and Copilot. International Journal of Scientific Research, Vol. 14, Issue 2. https://www.ijsr.net/getabstract.php?paperid=SR25207090345

[4] B. Yetistiren, I. Ozsoy, and E. Tuzun, “Assessing the quality of GitHub copilot’s code generation,” Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering, pp. 62–71, 2022, doi: https://doi.org/10.1145/3558489.3559072.

[5] R. Rajkumar, “Tabnine and Atlassian reveal new generative AI tools for developers,” ZDNET, 2024. https://www.zdnet.com/article/tabnine-and-atlassian-reveal-new-generative-ai-tools-for-developers/ (accessed 20)

[6] Singh, A., Mehta, R., & Banerjee, S. (2025) Comparative Evaluation of AI Code Assistants: Copilot, CodeWhisperer, and Tabnine. Axis Intelligence. Retrieved from https://axis-intelligence.com/ai-code-assistants-ultimate-comparison-2025/

[7] Mani, “Amazon CodeWhisperer —stepping on the accelerator ;-),” Medium, 2023. https://cmani.medium.com/amazon-codewhisperer-stepping-on-the-accelerator-1d0883ed649e (accessed 2024).

[8] Lee, J., & Nakamura, H. (2023). AI Coding Benchmark: Evaluating Performance of Leading AI Code Assistants Across Languages. AI Multiple Research. Retrieved from https://research.aimultiple.com/ai-coding-benchmark/

[9] Miller, T., et al. (2025) Generative AI in Education: Teaching Programming with Codeium and Ghostwriter. Journal of Educational Technology and Innovation, Vol. 18(1), pp. 22–34.

[10] Sandeep Shenoy., & Idoko Peter Idoko.,(2024) World Journal of Advanced Engineering Technology and Sciences