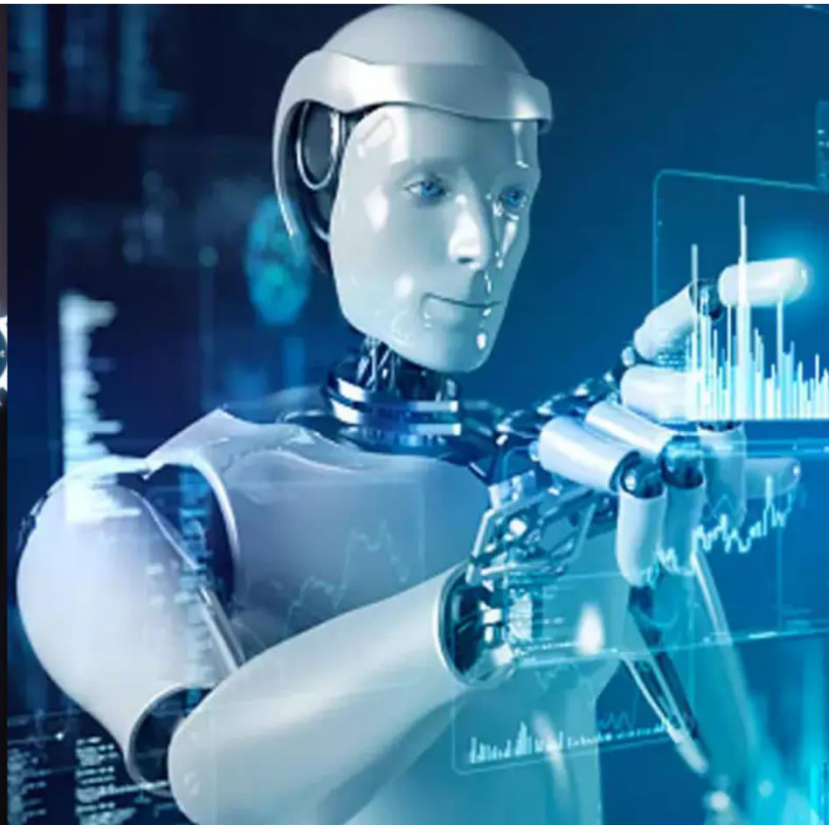
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 15, Issue 3, March 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 3, March 2026

|DOI: 10.15680/IJIRSET.2026.1503060|

Comparative Study of ChatGPT and Amazon Code Whisperer in Software Development

Ashwini Kapile

Assistant Professor, Ranibai Agnihotri Institute of Computer Science and Information Technology, Wardha, Maharashtra, India

ABSTRACT: Generative Artificial Intelligence (AI) is transforming software development by assisting developers throughout the coding process. This research presents a comparative study of ChatGPT and Amazon CodeWhisperer in software development tasks. The objective of this study is to evaluate their effectiveness, accuracy, efficiency, and usability in real-world programming scenarios. An experimental methodology was adopted in which both tools were tested on multiple programming tasks, including algorithm implementation, debugging, API integration, and documentation generation. The evaluation criteria included code correctness, response time, optimization quality, security suggestions, and ease of integration with development environments. The results indicate that ChatGPT performs well in generating detailed explanations, debugging support, and multi-language programming assistance. In contrast, Amazon CodeWhisperer demonstrates strong IDE integration and real-time code suggestions, particularly within cloud-based development environments. The findings suggest that both tools significantly improve developer productivity but differ in strengths depending on the development context. This study contributes to understanding how AI-powered coding assistants can enhance the Software Development Life Cycle (SDLC) and highlights the importance of responsible and effective adoption of generative AI technologies in modern software engineering practices.

KEYWORDS: Generative Artificial Intelligence (AI), Software Development Life Cycle (SDLC), AI Code Generation, ChatGPT, Amazon CodeWhisperer, Developer Productivity

I. INTRODUCTION

Generative Artificial Intelligence (AI) has emerged as a transformative force in modern software engineering, particularly in automating and augmenting various stages of the Software Development Life Cycle (SDLC). With the advancement of large-scale language models, AI systems are now capable of generating human-like text and functional programming code based on contextual prompts. These systems are built upon deep learning architectures trained on massive datasets containing both natural language and source code. The growing capability of such models has opened new possibilities for intelligent code assistance, automated debugging, and documentation generation (Brown et al., 2020). The foundation of modern generative AI systems lies in transformer-based neural networks, which enable models to process and generate sequential data efficiently. The transformer architecture, introduced by Vaswani et al. (2017), significantly improved contextual understanding in language tasks and later became the backbone of code-generating models. By leveraging attention mechanisms, these models can understand dependencies within long sequences of text or code, making them suitable for complex programming tasks. As a result, AI-powered coding assistants have gained popularity among developers seeking productivity enhancement.

Recent research indicates that large language models trained specifically on programming languages can perform competitive code synthesis and completion tasks. For instance, empirical evaluations demonstrate that AI systems can solve a considerable proportion of algorithmic programming problems, though performance depends on prompt quality and task complexity (Chen et al., 2021). These findings suggest that AI-driven tools can reduce development time by automating repetitive coding activities and supporting problem-solving processes. However, the effectiveness of such systems varies across different development environments and programming scenarios. Another significant development in this area is the introduction of instruction-tuned models, which improve alignment between human intent and machine-generated output. Instruction fine-tuning enhances model responsiveness and contextual understanding, thereby increasing usability in real-world applications (Ouyang et al., 2022). This advancement is

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 3, March 2026

|DOI: 10.15680/IJIRSET.2026.1503060|

particularly relevant in conversational AI tools that provide explanations alongside code generation, enabling developers to better understand and validate suggested solutions.

Despite their advantages, AI-assisted programming tools raise concerns regarding security, reliability, and ethical implications. Studies examining AI-generated code have identified potential security vulnerabilities, including insecure coding patterns and outdated library usage (Pearce et al., 2022). These findings highlight the necessity of human oversight and systematic evaluation when integrating AI into development workflows. Additionally, challenges such as intellectual property risks and bias in training data remain important considerations in responsible AI adoption. Research on open-source code generation models suggests that while generative AI can enhance productivity, it may also introduce inconsistencies in coding standards and maintainability (Fried et al., 2022). Therefore, understanding the comparative strengths and weaknesses of different AI coding assistants is essential for effective integration within the SDLC. Tools vary in their design philosophy—some emphasize conversational explanation and multi-language flexibility, while others focus on real-time IDE integration and cloud-based development support. Evaluating them based on parameters such as accuracy, efficiency, usability, security recommendations, and integration capability can provide valuable insights for developers and organizations. By systematically analyzing their performance in practical programming tasks, this research aims to contribute to a deeper understanding of how generative AI reshapes coding practices and influences software quality. Ultimately, while AI-driven development tools offer significant potential, their adoption must be guided by empirical evidence and careful assessment within professional software engineering environments.

II. REVIEW OF LITERATURE

1. Bommasani et al. (2021) – Bommasani and colleagues discussed the concept of foundation models and their broad applicability across domains, including software engineering. The study emphasized both the opportunities and systemic risks of large-scale AI models. It highlighted concerns related to reliability, bias, and governance while acknowledging their transformative potential in automating programming tasks and enhancing productivity within the Software Development Life Cycle (SDLC).

2. Nijkamp et al. (2022) – Nijkamp and co-authors introduced CodeGen, an open large language model designed specifically for code generation. Their research demonstrated that large-scale models trained on programming datasets can produce syntactically correct and context-aware code. The study evaluated performance across multiple programming languages and concluded that domain-specific training significantly improves code generation accuracy and adaptability in real-world applications.

3. Austin et al. (2021) – Austin and colleagues examined program synthesis capabilities of large language models. Their findings indicated that scaling model size and training data enhances performance in solving complex programming tasks. The research also noted that prompt engineering plays a crucial role in achieving optimal outputs, suggesting that effective human–AI interaction is essential for reliable integration in software development processes.

4. Li et al. (2022) – Li and co-authors conducted an empirical evaluation of AI-assisted coding tools to measure their impact on developer productivity. The study found that AI tools reduce coding time and support error correction but may introduce subtle logical inaccuracies. The authors recommended structured validation techniques to ensure code quality when integrating generative AI systems into professional development environments.

5. Xu et al. (2022) – Xu and colleagues proposed CodeGeeX, a multilingual code generation model. Their research demonstrated strong cross-language generalization capabilities and improved performance in automated programming benchmarks. The study emphasized the importance of diverse training datasets to enhance multi-language support, which is crucial for modern software projects that involve heterogeneous technology stacks.

6. Sandoval et al. (2023) – Sandoval and co-authors analyzed ethical and security concerns associated with AI-generated code. The study identified potential risks such as insecure coding practices, data leakage, and intellectual property violations. The authors concluded that while generative AI enhances efficiency, governance frameworks and human oversight are essential to ensure responsible and secure adoption in software engineering.

III. RESEARCH METHODOLOGY

This study adopts a comparative experimental research design to evaluate the performance of ChatGPT and Amazon CodeWhisperer in software development tasks. Both tools were tested under similar programming conditions to ensure fairness and consistency. The methodology focuses on assessing code accuracy, efficiency, usability, and security

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 3, March 2026

|DOI: 10.15680/IJIRSET.2026.1503060|

support across different stages of the Software Development Life Cycle (SDLC). The outputs generated by both systems were systematically analyzed and compared to identify their strengths and limitations.

1. Research Design: A comparative experimental approach was used. Both AI tools were given identical prompts and programming tasks to measure performance differences objectively.

2. Tools and Environment: The experiment was conducted using common programming languages such as Python and Java. Standard IDEs were used to observe real-time suggestions and integration capabilities.

3. Task Selection: Tasks included algorithm implementation, debugging, API integration, and documentation writing. These tasks represent real-world development activities within SDLC phases.

4. Evaluation Criteria: Performance was evaluated based on code correctness, response time, optimization quality, security recommendations, and ease of use. Results were recorded and compared using structured assessment metrics.

5. Experimental Setup: The experimental setup was designed to ensure a fair and structured comparison between ChatGPT and Amazon CodeWhisperer. Both tools were tested under similar development conditions to maintain consistency and reliability in results.

a. Development Environment: The experiment was conducted using standard Integrated Development Environments (IDEs) such as Visual Studio Code. Python and Java were selected as primary programming languages due to their widespread industry usage.

b. Hardware and System Configuration: Testing was performed on a standard computer system with stable internet connectivity. No special hardware acceleration was used to ensure realistic development conditions.

c. Task Design: A set of predefined programming tasks was created, including:

- Implementation of sorting and searching algorithms
- Debugging of logical and syntax errors
- Development of a simple REST API module
- Generation of code documentation and comments

Each task was executed separately using both AI tools.

d. Prompt Standardization: Identical prompts were provided to both tools to maintain fairness. Prompts were clearly structured to avoid ambiguity and ensure consistent input conditions.

e. Data Collection: Outputs generated by both tools were recorded and evaluated. Metrics such as code accuracy, execution success rate, response time, and quality of explanation were documented for comparison.

6. Validation Process: The generated code was compiled and executed to verify correctness. Manual review was conducted to check logical accuracy, security issues, and coding best practices.

IV. RESULTS AND ANALYSIS

The results of the experiment highlight notable differences in performance between ChatGPT and Amazon CodeWhisperer across various software development tasks. Both tools demonstrated strong capabilities in code generation, but their effectiveness varied depending on the nature of the task and development environment.

1. Code Accuracy: ChatGPT generated logically detailed and well-structured code in most algorithmic and conceptual tasks. It performed particularly well in explaining complex logic and providing step-by-step reasoning. CodeWhisperer, on the other hand, produced accurate real-time suggestions, especially for syntax completion and boilerplate code within the IDE. However, in some advanced logical implementations, minor corrections were required for both tools.

2. Response Time: CodeWhisperer provided faster inline code suggestions due to its direct integration with the development environment. ChatGPT required prompt submission and response generation, which slightly increased interaction time but offered more comprehensive outputs.

3. Code Optimization: ChatGPT often suggested optimized versions of algorithms and provided alternative approaches. CodeWhisperer focused more on immediate task completion rather than optimization improvements.

4. Debugging and Error Handling: ChatGPT showed strong debugging capability by identifying logical errors and suggesting corrective explanations. CodeWhisperer assisted in resolving syntax-level issues but offered limited detailed reasoning compared to ChatGPT.

5. Documentation and Explanation Quality: ChatGPT excelled in generating detailed documentation, comments, and conceptual explanations. CodeWhisperer generated concise comments but lacked in-depth explanatory detail.

6. Security Recommendations: Both tools provided basic security suggestions. However, security-specific improvements were more explicit in ChatGPT responses when directly prompted.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 3, March 2026

|DOI: 10.15680/IJIRSET.2026.1503060|

V. DISCUSSION

The findings of this study highlight the evolving role of generative AI tools in modern software development. The comparison between ChatGPT and Amazon CodeWhisperer demonstrates that both tools significantly support developers, but their effectiveness depends on task type and usage context. While both systems improve coding speed and reduce repetitive effort, their functional strengths differ in practical scenarios. ChatGPT proved highly effective in conceptual problem-solving, algorithm design, debugging, and documentation generation. Its conversational nature allows developers to ask follow-up questions, request explanations, and refine outputs iteratively. This makes it particularly useful during the requirement analysis, learning, and maintenance phases of the Software Development Life Cycle (SDLC). However, it operates outside the IDE environment, which may slightly interrupt workflow during rapid development tasks.

Amazon CodeWhisperer demonstrated strong performance in real-time code completion and seamless IDE integration. It efficiently generated boilerplate code and syntax-level suggestions, making it suitable for development and implementation phases. Its contextual suggestions within the coding environment enhance workflow continuity. However, it provided comparatively limited explanatory support and deeper logical reasoning. Another important observation is the need for human validation. Although both tools generated mostly accurate code, occasional logical errors, suboptimal implementations, and limited security considerations were identified. This confirms that AI tools should function as assistive technologies rather than autonomous developers. Over-reliance without verification may introduce hidden bugs or vulnerabilities. The discussion indicates that selecting an AI coding assistant depends on developer needs. For learning, debugging, and detailed reasoning, ChatGPT may be more beneficial. For rapid code generation and IDE-based assistance, CodeWhisperer may be preferable. A hybrid approach combining both tools could maximize productivity while maintaining software quality.

VI. ETHICAL AND SECURITY CONSIDERATIONS

The integration of generative AI tools such as ChatGPT and Amazon CodeWhisperer into software development introduces important ethical and security concerns that must be carefully addressed.

1. Code Security Risks: AI-generated code may contain hidden vulnerabilities, insecure libraries, or outdated practices. If developers rely on generated code without proper review, it can lead to security breaches, data leaks, or system failures. Therefore, manual code review and security testing remain essential.

2. Intellectual Property (IP) Issues: Generative AI models are trained on large datasets that may include publicly available source code. There is a potential risk of reproducing copyrighted or licensed code unintentionally. Organizations must ensure compliance with software licensing and legal standards.

3. Data Privacy Concerns: When developers input proprietary or confidential code into AI tools, there is a risk of exposing sensitive information. Proper data handling policies and secure usage guidelines should be followed to protect organizational data.

4. Bias and Fairness: AI systems may reflect biases present in their training data. This could influence coding patterns, documentation language, or design recommendations, potentially leading to unfair or inefficient outcomes.

5. Over-Reliance on AI: Excessive dependence on AI tools may reduce critical thinking and problem-solving skills among developers. Human expertise is necessary to validate logic, optimize performance, and ensure high-quality software.

6. Accountability and Responsibility: If AI-generated code causes system errors or security incidents, determining accountability becomes challenging. Clear governance policies and responsible AI usage frameworks should be established within organizations.

While generative AI tools enhance productivity and efficiency, ethical awareness, human oversight, and strict security validation are crucial to ensure responsible adoption in software engineering.

VII. CONCLUSION

The comparative study of ChatGPT and Amazon CodeWhisperer demonstrates that generative AI tools are significantly transforming modern software development practices. Both tools enhance developer productivity, reduce coding time, and assist in various stages of the Software Development Life Cycle (SDLC). The findings indicate that ChatGPT is particularly effective in providing detailed explanations, debugging support, optimized solutions, and comprehensive

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 3, March 2026

|DOI: 10.15680/IJIRSET.2026.1503060|

documentation. In contrast, Amazon CodeWhisperer excels in real-time code suggestions and seamless integration within Integrated Development Environments (IDEs), making it highly efficient for rapid development tasks. However, minor inaccuracies and security considerations highlight the need for human validation and oversight. Overall, while both AI tools offer substantial benefits, their effectiveness depends on the specific development context and user requirements. Therefore, integrating generative AI into software engineering should be approached as a collaborative human–AI partnership rather than a complete replacement for developers.

VIII. FUTURE SCOPE

The future scope of this research lies in expanding the comparative evaluation of ChatGPT and Amazon CodeWhisperer across larger and more complex real-world software projects. Future studies can include additional programming languages, frameworks, and enterprise-level applications to measure scalability and long-term performance. Quantitative metrics such as productivity rate, defect density, and code maintainability can also be analyzed using larger datasets. Further research may explore the integration of generative AI tools with DevOps practices, cloud-native development, and automated testing pipelines. Security-focused evaluation can be conducted to assess vulnerability detection and compliance with secure coding standards. Additionally, studying human–AI collaboration models and the impact of AI assistance on developer learning and decision-making would provide deeper insights. As AI models continue to evolve, future work can also compare newer versions and emerging AI coding assistants to understand technological advancements and their implications for the Software Development Life Cycle (SDLC).

REFERENCES

1. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.
2. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., … Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374. https://arxiv.org/abs/2107.03374
3. Fried, D., Aghajanyan, A., Lin, J., Wallace, E., Shi, F., Yih, W., … Zettlemoyer, L. (2022). InCoder: A generative model for code infilling and synthesis. arXiv preprint arXiv:2204.05999. https://arxiv.org/abs/2204.05999
4. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., … Lowe, R. (2022). Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35, 27730–27744.
5. Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. IEEE Symposium on Security and Privacy Workshops (SPW), 1–9.
6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.
7. Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., … Sutton, C. (2021). Program synthesis with large language models. arXiv preprint arXiv:2108.07732. https://arxiv.org/abs/2108.07732
8. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., … Liang, P. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258. https://arxiv.org/abs/2108.07258
9. Li, Y., Wang, S., & Yin, M. (2022). An empirical study of AI-assisted programming tools and their impact on developer productivity. IEEE Transactions on Software Engineering.
10. Nijkamp, E., Hayashi, H., Xie, C., & Liang, P. (2022). CodeGen: An open large language model for code generation. arXiv preprint arXiv:2203.13474. https://arxiv.org/abs/2203.13474
11. Sandoval, G., Pearce, H., & Karri, R. (2023). Security implications of AI-generated code in software development. IEEE Software, 40(3), 45–52.
12. Xu, Y., Li, Z., Li, Y., Gao, R., & others. (2022). CodeGeeX: A multilingual code generation model. arXiv preprint arXiv:2203.17568. https://arxiv.org/abs/2203.17568

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details