

A Survey-Based Study on the Adoption of Generative AI in Software Development: Opportunities, Challenges and User Perspectives

Rutuja Rajendra Yadav, Rushikesh Ganesh Yadav, Pratiksha Kamble

Zeal Institute of Business Administration, Computer Application and Research (ZIBACAR), Pune, Maharashtra, India

Published October 2026

Abstract

Generative AI (GenAI) is increasingly used in software development for coding, debugging, documentation, and learning. This study examines its adoption, benefits, challenges, security concerns, dependence, learning support, and future impact. Survey responses from 21 programming students and professionals were analyzed using frequencies, percentages, and average ratings. Of the respondents, 95.2% reported daily or frequent use and 95.2% used ChatGPT. Debugging and error fixing was the most common task (71.4%). Respondents reported benefits such as better problem-solving and faster development, but also identified bugs, security issues, architectural complexity, and privacy concerns. Overall, GenAI is viewed as a useful assistant, but human review, testing, and responsible use remain essential. The findings are exploratory due to the small sample size.

Keywords: generative AI; software development; ChatGPT; AI-assisted programming; code generation; user perspectives

1 Introduction

Generative AI (GenAI) refers to AI systems that can create content such as text, code, explanations, and documentation. Tools like ChatGPT, GitHub Copilot, Claude, and Google Gemini support programmers in coding, debugging, learning, and problem-solving.

GenAI allows users to describe requirements in natural language and receive suggested solutions, reducing repetitive work and supporting unfamiliar tasks. However, AI-generated code may contain bugs, security issues, or incomplete logic and therefore requires human verification.

This study examines how programming students and professionals use GenAI, its benefits and limitations, and concerns related to security, dependence, verification, learning, and the future of software-development jobs.

2 Background and literature review

Previous research shows that Generative AI can improve productivity and support software development, but human judgment remains important. [Noy and Zhang \(2023\)](#) reported productivity benefits from GenAI, while [Barke et al. \(2023\)](#) showed that developers actively guide and evaluate AI-generated suggestions. [Vaithilingam et al. \(2022\)](#) highlighted the importance of understanding users' actual experience with AI coding tools.

Pearce et al. (2022) identified security weaknesses in AI-generated code, showing that generated code requires proper review and testing. Overall, research indicates that GenAI can support coding, debugging, and learning, but human verification, security awareness, and responsible use remain essential.

3 Research gap

Existing studies discuss the capabilities, productivity effects, usability, and security implications of AI-assisted programming. However, there is value in examining the combined user perspective across adoption frequency, tools used, development tasks, perceived code quality, benefits, limitations, privacy concerns, dependence, learning, verification, and future job expectations. The present study addresses this gap through a small survey of programming students and professionals and provides an exploratory snapshot of their experiences.

4 Problem statement

The increasing use of Generative AI in software development provides opportunities for faster coding, debugging, learning, and problem-solving. At the same time, users may face inaccurate or insecure generated code, difficulty applying AI to complex architecture, privacy concerns, and over-dependence. Therefore, there is a need to understand how programming-related users adopt GenAI, what benefits they experience, what challenges they face, and how they perceive its role in the future of software development.

5 Research objectives

1. To study the adoption and use of Generative AI tools in software development.
2. To identify the major opportunities and benefits of using Generative AI for programming.
3. To identify the challenges and concerns faced while using Generative AI in software development.
4. To analyse users' perspectives regarding the usefulness, accuracy, and reliability of GenAI tools.
5. To analyse survey responses and identify common usage patterns and perceptions.

6 Research methodology

6.1 Research design

The study follows a quantitative, descriptive, survey-based research design. Primary data were collected using a structured Google Forms questionnaire. The questionnaire focused on respondent background, frequency of GenAI use, tools used, software-development tasks, perceived quality, benefits, challenges, security and privacy, dependence, time savings, learning, verification, future job impact, and recommendation.

6.2 Sample and data collection

A total of 21 valid responses were available in the supplied dataset. The respondents included MCA students, Computer Science students/graduates, programming professionals, and one professional software developer. The sample was obtained through the survey instrument and represents a

convenience/voluntary response sample; therefore, it should be interpreted as an exploratory sample rather than a statistically representative sample of all developers.

6.3 Data analysis

Responses were analysed using frequency counts, percentages, and mean ratings where numerical scales were used. For multiple-response questions, each selected option was counted independently; consequently, percentages for such questions can sum to more than 100%. Charts were prepared to make the major patterns easier to interpret.

6.4 Research process

The research followed these steps: questionnaire preparation → data collection → data cleaning → descriptive analysis → charts/tables → interpretation → findings → conclusion.

7 Data analysis and results

7.1 Respondent profile

The largest group was MCA students (11; 52.4%), followed by other programming professionals (5; 23.8%) and Computer Science graduates/students (4; 19.0%). This indicates that the sample is particularly relevant to the student and early-career programming perspective. The responses are summarized in Table 1.

Table 1. Respondent profile ($N = 21$).

Role / background	n	%
MCA Student	11	52.4
Other Programming Professional	5	23.8
Computer Science Graduate/Student	4	19.0
Professional Software Developer	1	4.8

7.2 Frequency of GenAI use

Daily use was reported by 12 respondents (57.1%), while 8 (38.1%) reported frequent use. Together, 20 respondents (95.2%) use GenAI daily or frequently. Only one respondent reported occasional use. This indicates a high level of adoption within the sample. The responses are summarized in Table 2.

Table 2. Frequency of GenAI use ($N = 21$).

Frequency	n	%
Daily	12	57.1
Frequently	8	38.1
Occasionally	1	4.8

7.3 Generative AI tools used

ChatGPT was the most frequently selected tool, used by 20 respondents (95.2%). Google Gemini was selected by 13 (61.9%), Claude by 11 (52.4%), and GitHub Copilot by 7 (33.3%). Because respond-

ents could select multiple tools, these percentages represent tool usage rather than exclusive categories. The responses are summarized in Table 3.

Table 3. Generative AI tools used ($N = 21$).

Tool	n	%
ChatGPT	20	95.2
Google Gemini	13	61.9
Claude	11	52.4
GitHub Copilot	7	33.3

Multiple responses allowed; percentages sum to more than 100%.

7.4 Software development tasks supported by GenAI

Debugging and error fixing was the most common use, selected by 15 respondents (71.4%). Writing new code, code explanation/documentation, and learning new programming languages or concepts were each selected by 11 respondents (52.4%). Unit-test writing was selected by only two respondents, suggesting comparatively lower use for that task in this sample. The responses are summarized in Table 4.

Table 4. Development tasks supported by GenAI ($N = 21$).

Task	n	%
Debugging and error fixing	15	71.4
Writing new code	11	52.4
Code explanation and documentation	11	52.4
Learning new programming languages or concepts	11	52.4
Writing unit tests	2	9.5

Multiple responses allowed; percentages sum to more than 100%.

7.5 Perceived quality and accuracy of AI-generated code

The mean quality/accuracy rating was 3.62 out of 5. A rating of 4 was given by 12 respondents (57.1%). Ratings of 3 or above were given by 18 respondents (85.7%). This indicates generally positive perceptions, but not universal confidence in AI-generated code. The responses are summarized in Table 5.

Table 5. Perceived quality and accuracy of AI-generated code (rating 1–5) ($N = 21$).

Rating	n	%
2	3	14.3
3	4	19.0
4	12	57.1
5	2	9.5

7.6 Benefits of using GenAI

Better problem-solving assistance was the most common benefit (14 respondents; 66.7%), followed by increased development speed (13; 61.9%). Reduced routine work and an easier learning curve were also frequently reported. The results suggest that respondents value GenAI both as a productivity aid and as a learning/problem-solving assistant. The responses are summarized in Table 6.

Table 6. Benefits of using GenAI ($N = 21$).

Benefit	n	%
Better problem-solving assistance	14	66.7
Increased development speed	13	61.9
Reduced routine and repetitive tasks	10	47.6
Easier learning curve for new technologies	10	47.6
Improved code structure	8	38.1

Multiple responses allowed; percentages sum to more than 100%.

7.7 Challenges and limitations

The most frequently reported challenge was that generated code may contain bugs or security flaws (14 respondents; 66.7%). Difficulty with complex architectural logic was next (13; 61.9%). These findings reinforce the idea that GenAI output is useful as assistance but requires contextual understanding and validation. The responses are summarized in Table 7.

Table 7. Challenges and limitations ($N = 21$).

Challenge	n	%
Generated code contains bugs or security flaws	14	66.7
Difficulty with complex architectural logic	13	61.9
Lack of understanding of the underlying codebase	10	47.6
Privacy and data security concerns	8	38.1
Over-reliance leading to reduced problem-solving skills	6	28.6

Multiple responses allowed; percentages sum to more than 100%.

7.8 Security and privacy concerns

Fifteen respondents (71.4%) were at least moderately concerned about security and privacy, while 6 respondents were neutral. This indicates that data protection remains a significant issue for users considering AI-assisted programming. The responses are summarized in Table 8.

Table 8. Security and privacy concerns ($N = 21$).

Concern level	n	%
Neutral	6	28.6
Moderately Concerned	10	47.6
Extremely Concerned	5	23.8

7.9 Perceived dependence on GenAI

Nine respondents (42.9%) reported significant dependence and five (23.8%) reported some dependence. Thus, 14 respondents (66.7%) perceived at least some dependence. This is important because convenience and productivity gains may also create a risk of reduced independent problem-solving if AI is used without sufficient understanding. The responses are summarized in Table 9.

Table 9. Perceived dependence on GenAI ($N = 21$).

Dependence	n	%
Not very dependent	2	9.5
Neutral	5	23.8
Somewhat dependent	5	23.8
Yes, significantly dependent	9	42.9

7.10 Satisfaction with time saved

The mean time-saving satisfaction rating was 3.62 out of 5. Ratings of 4 or 5 were given by 12 respondents (57.1%), indicating that most respondents experienced noticeable time-saving benefits, although a small number reported low satisfaction. The responses are summarized in Table 10.

Table 10. Satisfaction with time saved (rating 1–5) ($N = 21$).

Rating	n	%
1	2	9.5
2	1	4.8
3	6	28.6
4	6	28.6
5	6	28.6

7.11 Support for learning complex algorithms and code patterns

GenAI was reported to help with understanding complex algorithms and code patterns always by 9 respondents (42.9%) and often by 7 (33.3%). Therefore, 16 respondents (76.2%) reported that GenAI often or always supports learning. The responses are summarized in Table 11.

Table 11. Support for learning complex algorithms and code patterns ($N = 21$).

Response	n	%
Sometimes	5	23.8
Often	7	33.3
Always	9	42.9

7.12 Independent review, testing and verification

Thirteen respondents (61.9%) review, test, and verify AI-generated code most of the time, while six (28.6%) do so always. Only two respondents (9.5%) selected sometimes. This is a positive practice

because it shows that most respondents do not treat AI-generated output as automatically reliable. The responses are summarized in Table 12.

Table 12. Independent review, testing and verification of AI-generated code ($N = 21$).

Verification frequency	n	%
Sometimes	2	9.5
Most of the time	13	61.9
Always	6	28.6

7.13 Perceived impact on future software-development jobs

Twelve respondents (57.1%) believe GenAI will significantly transform developer roles, while five (23.8%) believe it will completely replace developers. Only four (19.0%) expect it to remain a minor assistant tool. The responses therefore show strong expectations of substantial change in software-development work. The responses are summarized in Table 13.

Table 13. Perceived impact on future software-development jobs ($N = 21$).

Perceived impact	n	%
Will serve as a minor assistant tool	4	19.0
Will significantly transform developer roles	12	57.1
Will completely replace developers	5	23.8

7.14 Recommendation to students and junior developers

Eleven respondents (52.4%) recommend adoption and four (19.0%) strongly recommend it. Thus, 15 respondents (71.4%) recommend GenAI to students and junior developers, while six respondents remain neutral. No respondent selected a negative recommendation in the supplied response categories. The responses are summarized in Table 14.

Table 14. Recommendation to students and junior developers ($N = 21$).

Recommendation	n	%
Neutral	6	28.6
Recommend	11	52.4
Strongly Recommend	4	19.0

7.15 Summary of key findings

Figure 1 summarises selected indicators drawn from the tables above. Each bar comes from a separate survey question.

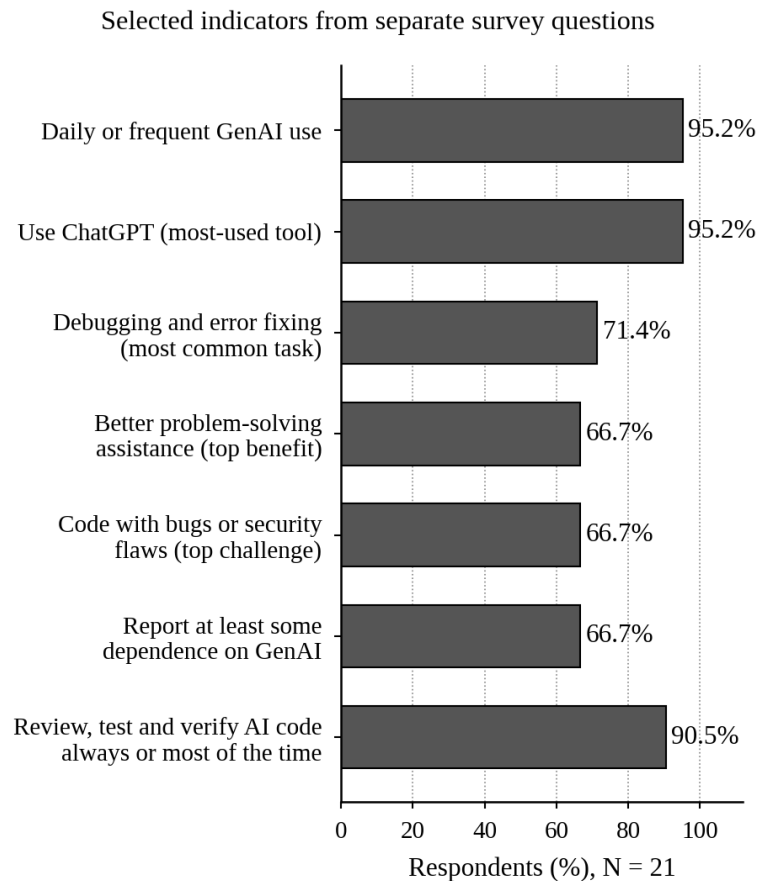


Figure 1. Summary of key findings on generative AI adoption in software development ($N = 21$). Bars show selected indicators from separate survey questions; they are not mutually exclusive categories or parts of a single total.

8 Discussion

The survey shows that GenAI has become part of regular coding and academic work for most respondents, with ChatGPT being the most used tool. Debugging is the most common application, showing the usefulness of AI for identifying and solving programming problems.

However, generated bugs and security flaws were also the most reported limitation. The average ratings for code quality and time savings were both 3.62/5, indicating positive but not complete trust.

Dependence is also important, with 66.7% reporting some level of dependence, while 90.5% reported regularly verifying AI-generated code. This shows that users may rely on AI for assistance while still maintaining human review.

Overall, respondents expect GenAI to significantly transform software-development roles. The findings suggest that GenAI is a useful development assistant, but human judgment, verification, and technical skills remain essential.

9 Implications of the study

- *Education:* Use GenAI as a learning assistant while requiring students to understand and test generated code.
- *Development:* Treat AI output as a suggestion and perform proper code review, testing, and security checks.

-
- *Organizations*: Establish clear policies for AI use, privacy, and proprietary code.
 - *Training*: Focus on verification, debugging, architecture, and independent problem-solving along with prompt writing.

10 Limitations

- *Small sample*: Only 21 responses were collected, limiting generalization.
- *Sample bias*: MCA students form a large part of the sample.
- *Self-reported data*: Responses may reflect personal perceptions or recall.
- *Descriptive study*: The research does not establish causal relationships.

11 Conclusion

This study examined GenAI adoption in software development based on 21 responses. GenAI is frequently used, especially ChatGPT, for debugging, coding, explanation, and learning. Respondents reported benefits such as faster development and better problem-solving, with an average rating of 3.62/5 for code quality and time savings.

However, bugs, security, privacy, architectural complexity, and AI dependence remain important concerns. Overall, GenAI is best viewed as an assistant rather than a replacement for human programming skills and judgment. Larger studies are needed to validate these findings.

12 Future scope

- Conduct surveys with larger and diverse groups.
- Compare GenAI use across experience levels and programming domains.
- Compare AI tools based on usefulness, accuracy, and security.
- Study changes in productivity, dependence, and coding skills over time.
- Evaluate actual coding performance alongside user perceptions.
- Study organizational GenAI policies and data-privacy practices.

Appendix A: Survey variables

- Respondent background
- GenAI usage frequency
- AI tools used
- Development tasks
- Code quality
- Benefits
- Challenges
- Security and privacy
- AI dependence
- Time savings
- Learning support
- Code verification

-
- Future job impact
 - Recommendation of GenAI

References

- [1] Barke S, James MB, Polikarpova N (2023). Grounded Copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA1), 85–111. <https://doi.org/10.1145/3586030>
- [2] GitHub (2024). GitHub Copilot documentation. *GitHub*.
- [3] Google (2024). Gemini documentation. *Google*.
- [4] Noy S, Zhang W (2023). Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654), 187–192. <https://doi.org/10.1126/science.adh2586>
- [5] OpenAI (2023). GPT-4 technical report. *arXiv:2303.08774*.
- [6] 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. In: *2022 IEEE Symposium on Security and Privacy (SP)*, 754–768. <https://doi.org/10.1109/SP46214.2022.9833571>
- [7] Peng S, Kalliamvakou E, Cihon P, Demirer M (2023). The impact of AI on developer productivity: Evidence from GitHub Copilot. *arXiv:2302.06590*.
- [8] Vaithilingam P, Zhang T, Glassman EL (2022). Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In: *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 1–7. <https://doi.org/10.1145/3491101.3519665>