ARTIFICIAL INTELLIGENCE AND GENERATIVE AI

The Trustworthiness of AI-Generated Code: A Survey of How Students and Working Professionals Trust, Check and Use Code Written by Generative AI

Vrushali Ghodase, Vidya Salunke, Mansi Sawant, Sanika Sawant

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

Published November 2026

Abstract

Generative AI tools such as ChatGPT and GitHub Copilot now produce a growing share of the code used by students and working professionals, yet earlier studies show that such code can be incorrect, insecure or difficult to maintain. This paper examines how much people trust AI-generated code, whether they check it before use, and how closely the two are aligned. A descriptive, quantitative survey was run with a structured Google Forms questionnaire distributed through convenience sampling between 8 and 25 October 2026, and 16 valid responses from people who use AI for coding were analysed using frequencies, percentages and mean scores. As its main contribution, the paper proposes a Verification Gap (VG), defined as the percentage of respondents with high trust minus the percentage with high verification, together with a four-group Trust-Verification Matrix that classifies users as calibrated, at-risk, cautious or disengaged. Of the respondents, 68.8% use AI for coding daily, 68.8% report high trust in the code and 56.3% both check and test it, giving a VG of 12.5 percentage points; 18.8% are at-risk users who trust the code without both checking and testing it. Perceived correctness (mean 4.25 of 5) exceeded perceived safety (mean 3.63), and 37.5% of respondents had met security issues. The findings are interpreted against published evidence on correctness, security and developer trust. Because they rest on a very small, self-reported, non-probability sample, they are exploratory and cannot be generalised.

Keywords: AI-generated code; code verification; developer trust; generative AI; software security; survey; trustworthiness; Verification Gap

1 Introduction

1.1 Background of the study

Until recently, writing a program meant that its author understood every line of it. Large language models trained on public source code have changed this: a short instruction in plain language can now return a working function, a database query or an entire web page. Tools such as GitHub Copilot and ChatGPT are used in assignments, mini-projects, internships and professional work, and research on them commonly relies on benchmarks that run generated programs against unit tests [1].

Speed, however, is not the same as trustworthiness. Code that compiles can still give wrong results for unusual inputs, contain security weaknesses or be hard to maintain. Studies of AI code assistants report that a large share of suggestions in security-relevant scenarios were vulnerable [2], that participants who had an assistant wrote less secure solutions while feeling more confident about them [3], and that code from more than one hundred models contained known web-application weaknesses in

45% of test cases [4]. Developers are themselves wary: in a large 2025 survey, more respondents distrusted the accuracy of AI tools than trusted them [5].

1.2 Problem statement

Trustworthiness has a technical side (is the code correct and safe?) and a human side (does the user believe it is, and does the user check?). When belief is high and checking is low, defects pass unnoticed into assignments and products. Little is known about how learners and early-career users in India describe their own trust and checking habits in everyday use of AI-generated code. This study therefore asks three linked questions: how much do respondents trust AI-generated code, how do they check it, and does their level of trust match their level of checking?

1.3 Research objectives

The main objective is to examine the trust that students and working professionals place in AI-generated code and the checking practices that accompany it. The specific objectives are:

1. To describe the frequency, tools and purposes of AI use for coding.
2. To measure perceived correctness, safety and overall trust in AI-generated code.
3. To study whether respondents check and test AI-generated code before using it.
4. To identify the problems respondents have actually met in AI-generated code.
5. To propose and compute a Verification Gap and a Trust-Verification Matrix that show whether trust is matched by checking.

1.4 Significance of the study

The study offers teachers and institutions a small piece of evidence on where checking habits are strong or weak, which can inform policies on responsible AI use in programming courses. For researchers, it adds a survey-based view that complements laboratory studies of code quality, and it provides a simple, reusable measure of trust calibration.

The remainder of the paper reviews the literature and identifies the research gap, describes the methodology, including the proposed measures, presents and discusses the results, and then sets out the findings and recommendations, limitations, future scope and conclusion.

2 Review of literature

2.1 Correctness of AI-generated code

Chen et al. [1] introduced the Codex model and the HumanEval benchmark, in which generated programs are judged by unit tests rather than by similarity to a reference solution. Their model solved 28.8% of the problems with a single attempt, and repeated sampling at 100 attempts per problem raised the reported share of problems solved to 70.2%. Two points follow for the present study: correctness has to be established by running the code, and the same prompt can yield both working and failing answers.

2.2 Security of AI-generated code

Pearce et al. [2] prompted GitHub Copilot with 89 scenarios built around high-risk weakness types and found that about 40% of the 1,689 resulting programs were vulnerable. The 2025 Veracode re-

port [4] set 80 tasks for more than 100 models and found weaknesses from the OWASP Top 10 in 45% of cases. Java had the highest failure rate, above 70%, and cross-site scripting and log injection were poorly defended (failure rates of 86% and 88%). These results make security a necessary part of any assessment of trust.

2.3 Trust, confidence and over-reliance

Perry et al. [3] conducted a user study in which participants with an AI assistant wrote less secure code on most tasks, yet were more likely to believe that their code was secure. Kasneci et al. [6] discuss the opportunities and risks of large language models in education, including over-reliance and the need to preserve critical thinking. Fluent but ungrounded output, usually called hallucination, is a recognised property of generative models [7]. In the Stack Overflow 2025 survey [5], 46% of developers distrusted the accuracy of AI tools, 33% trusted it and only 3% trusted it highly, and many described AI answers as almost right but in need of extra debugging.

2.4 Research gap

Earlier work either measures the code itself (its correctness or security) or measures attitudes (trust), and it rarely relates trust to checking for the same individuals. Survey evidence from Indian learners and early-career users is also limited. This study supplies a small, survey-based description of trust and checking, and it introduces measures that join the two.

3 Research methodology

3.1 Research design

The study follows a descriptive, quantitative survey design [8]. It characterises respondents' use of AI for coding, their trust, their checking and testing practice, and the problems they have met. All findings are self-reported; no code quality was measured and no usage logs were collected.

3.2 Population, sample and sampling

The target population consists of people who use AI tools to help with programming or coding. Respondents were selected by convenience sampling: the questionnaire was shared through personal and online contacts between 8 and 25 October 2026, and people who were accessible and willing completed it voluntarily. In total, 20 responses were received. Three respondents who had never used AI for coding were excluded as outside the target population, and one response from a person under 18 was excluded for ethical reasons, leaving N = 16 valid responses. Three of these 16 were completed by the authors; results without them are reported as a check in the Results section. Because the sample is small and non-probabilistic, the results describe these respondents only.

3.3 Research instrument

Primary data were collected with a structured Google Forms questionnaire of 19 closed-ended questions, supplemented by name and email fields that are not reported. The questions cover (A) age, occupation and programming experience; (B) AI tools, frequency and purpose of use; (C) perceived correctness, working, safety and fit to requirements of AI-generated code; (D) checking and testing before use; (E) understanding, confidence, the effect of past experience and willingness to use the code in a real project; and (F) problems faced and overall trust. Ten statements and the overall trust

question were rated on a five-point scale from 1 (strongly disagree, or no trust) to 5 (strongly agree, or complete trust). The full questionnaire is given in the Appendix.

3.4 Data collection and analysis

Primary data came from the Google Forms responses; secondary sources (research papers, reports and official publications) formed the basis of the Introduction and the Review of Literature. Responses were tabulated and analysed using frequency counts, percentages and mean scores. Percentages were calculated as

$$p_i = \frac{f_i}{N} \times 100 \tag{1}$$

where f_i is the frequency of response category i, N is the number of valid responses and p_i is the corresponding percentage. A respondent is counted as agreeing with a statement if the rating is 4 or 5. Percentages and means are rounded to one and two decimal places respectively, and differences between percentages are computed from the unrounded values. Spearman's rank correlation is reported only as a descriptive measure; with N = 16, no claim of statistical significance is made for any comparison.

3.5 Proposed Verification Gap and Trust-Verification Matrix

This subsection defines the measures proposed as the research contribution. Trust and checking are each reduced to a high or a low category. A respondent has High Trust (HT) if the overall trust rating is 4 or 5. A respondent has High Verification (HV) if both the rating for checking before use and the rating for testing before use are 4 or 5. The Verification Gap is then defined as

$$\mathrm{VG} = \%\mathrm{HT} - \%\mathrm{HV} \tag{2}$$

where %HT and %HV are the percentages of all respondents in the high-trust and high-verification categories. A positive VG means that trust exceeds checking, a value near zero indicates balance, and a negative value indicates that checking exceeds trust.

To describe individuals, every respondent is placed in a Trust-Verification Matrix of four groups: Calibrated (HT and HV), At-risk (HT but not HV), Cautious (not HT but HV) and Disengaged (neither). The At-risk group is the main concern, because its members rely on code that they have not both checked and tested. A security-specific gap is obtained by replacing overall trust with agreement (rating 4 or 5) that AI-generated code is safe to use:

$$\mathrm{VG}_s = \%S - \%\mathrm{HV} \tag{3}$$

where %S is the percentage of respondents who agree that AI-generated code is safe to use.

4 Results and analysis

4.1 Profile of respondents

Table 1 summarises the profile of the 16 valid respondents. They were mostly young adults, with 62.5% aged 21–25. Students formed 31.3% of the sample, software developers or engineers 37.5%, and other professionals (such as analysts and managers) 31.3%. Half of the respondents had less than one year of programming experience, so the sample leans toward new and occasional programmers.

Table 1. Profile of respondents (N = 16).

Characteristic	n	%
Age group		
21–25 years	10	62.5
26–30 years	2	12.5
31–40 years	2	12.5
Above 40 years	2	12.5
Occupation		
Student	5	31.3
Software developer / engineer	6	37.5
Other professional (analyst, manager, etc.)	5	31.3
Programming experience		
Less than 1 year	8	50.0
1–2 years	2	12.5
3–5 years	4	25.0
More than 5 years	2	12.5

4.2 Use of AI for coding

All 16 respondents (100%) had used ChatGPT. GitHub Copilot, Google Gemini, Claude and Microsoft Copilot had each been used by 62.5%. Use was frequent: 68.8% used AI for coding daily, 12.5% several times a week, 6.3% sometimes and 12.5% rarely. The most common purpose was understanding code (87.5%), followed by writing code, finding and fixing errors, and learning programming (75.0% each), then writing SQL queries (43.8%) and testing code (37.5%).

4.3 Perceived quality, trust and checking

Table 2 reports the mean rating and the level of agreement for each statement. Respondents rated correctness (mean 4.25) and fit to requirements (4.00) highest, and safety lowest (3.63), among the quality items; two respondents disagreed that AI-generated code is safe to use. Checking (62.5%) and testing (56.3%) before use, together with perceived safety (62.5%), received the lowest agreement of all statements (Figure 1). Overall, 68.8% reported high trust.

Table 2. Ratings of AI-generated code (N = 16, scale 1–5).

Statement	Mean	Agree n	Agree %
Gives the correct solution	4.25	13	81.3
Works as expected	3.88	12	75.0
Safe to use	3.63	10	62.5
Meets my requirements	4.00	13	81.3
I check it before using	3.94	10	62.5
I test it before using	3.81	9	56.3

Statement	Mean	Agree n	Agree %
Easy to understand and modify	3.75	12	75.0
Confident using it after checking	4.00	12	75.0
Past experience affects my trust	3.94	12	75.0
Would use in a real project after checking	3.94	12	75.0
Overall trust in AI code	3.88	11	68.8

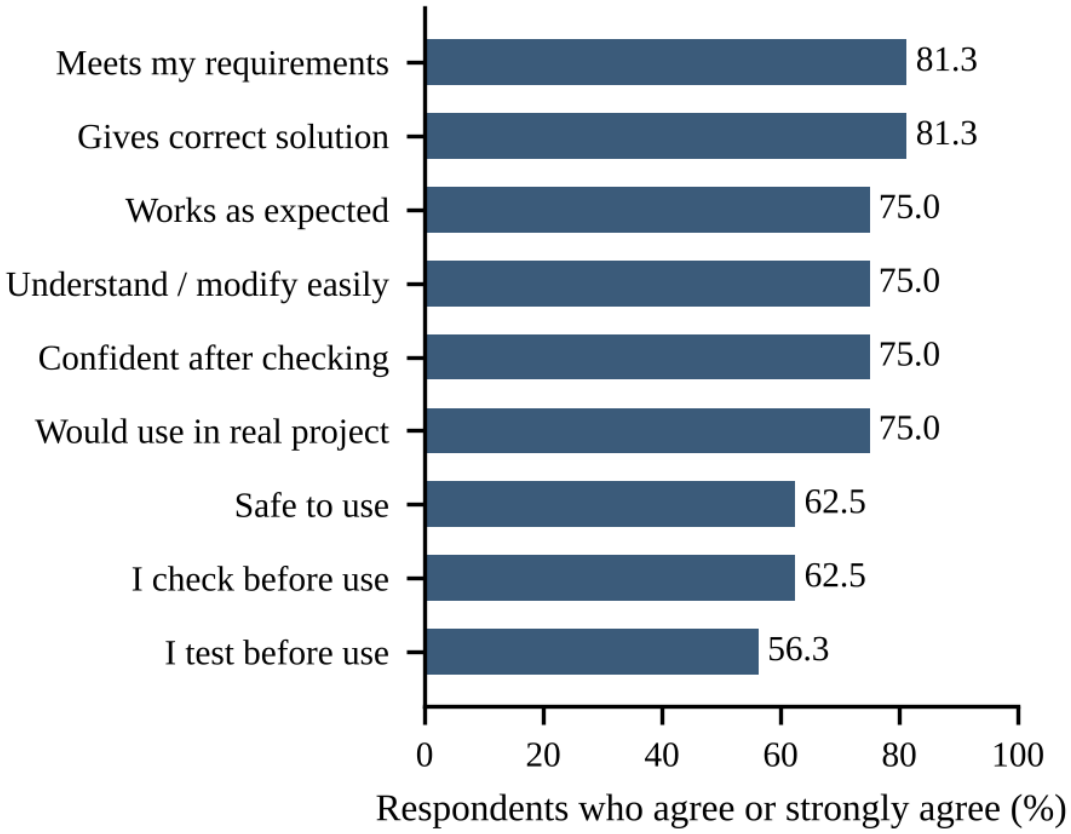


Figure 1. Agreement with statements about AI-generated code (% of respondents, N = 16).

Overall trust was strongly associated with perceived safety (Spearman's ρ = 0.76) and only weakly associated with the combined checking-and-testing score (ρ = 0.39). Both coefficients are descriptive; a sample of 16 does not support inference.

4.4 Problems faced

Table 3 lists the problems that respondents had met in AI-generated code. Errors or bugs were the most common (56.3%), followed by security issues and code that did not meet requirements (37.5% each). Respondents could select more than one option, so the percentages add to more than 100. Only 43.8% reported no major problems.

Table 3. Problems faced with AI-generated code (N = 16).

Problem	n	%
Errors or bugs	9	56.3
Security issues	6	37.5
Code did not meet requirements	6	37.5
Wrong logic	5	31.3
Incomplete code	5	31.3
Inefficient code	4	25.0
Wrong output	2	12.5
No major problems faced	7	43.8

4.5 Verification Gap and Trust-Verification Matrix

High trust was reported by 11 of the 16 respondents (%HT = 68.8) and high verification, meaning that both checking and testing were rated 4 or 5, by 9 of 16 (%HV = 56.3). By Equation (2), VG = 68.8 − 56.3 = 12.5 percentage points. For safety, 62.5% agreed that AI-generated code is safe to use, so by Equation (3) VG_s = 6.3 points (10/16 − 9/16 = 0.0625, shown to one decimal place). Table 4 and Figure 2 place each respondent in the Trust-Verification Matrix.

Table 4. Trust-Verification Matrix (N = 16).

Group	Definition	n	%
Calibrated	High trust, high verification	8	50.0
At-risk	High trust, not high verification	3	18.8
Cautious	Low trust, high verification	1	6.3
Disengaged	Low trust, not high verification	4	25.0

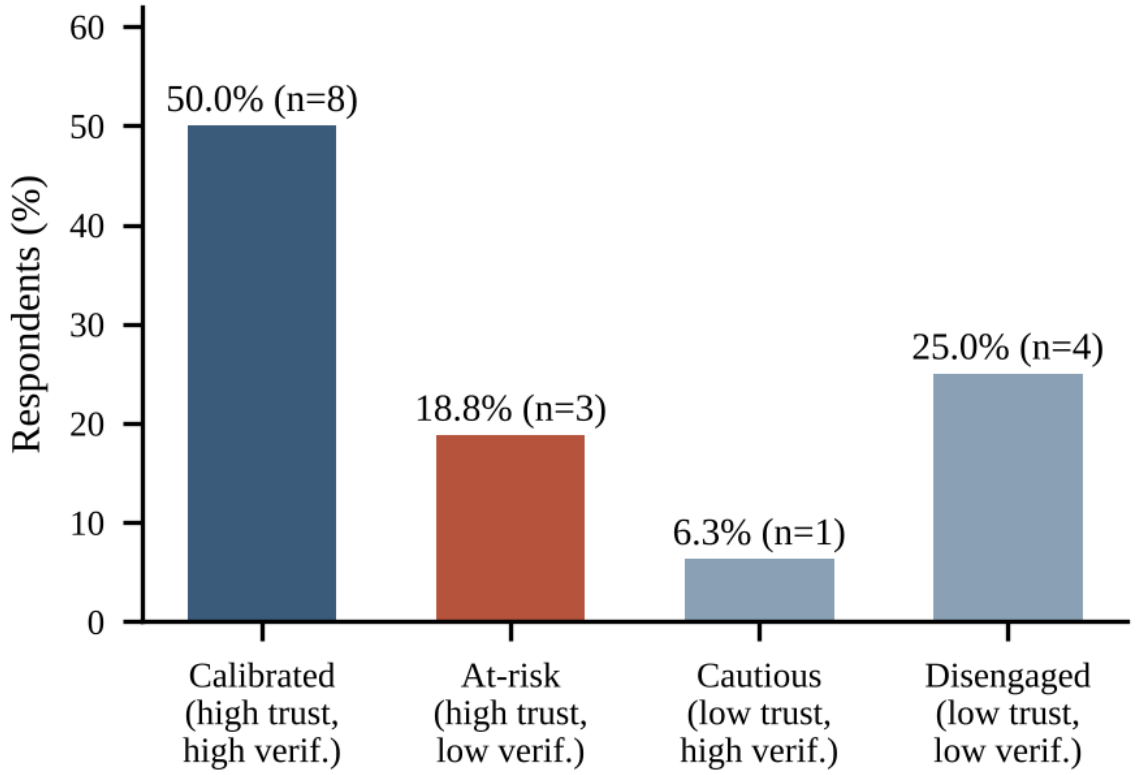


Figure 2. Trust-Verification Matrix groups (% of respondents, N = 16).

Half of the respondents are calibrated, and 18.8% (three respondents) are at-risk. Two of the three at-risk respondents have 3–5 years of experience, so experience did not remove the risk, although the numbers are too small to compare groups. When the three responses completed by the authors are removed (N = 13), %HT is 76.9, %HV is 61.5 and VG is 15.4 points, so the direction of the result does not change.

5 Discussion

For these respondents, AI is a routine coding aid: about seven in ten use it daily, and most use it to understand, write and fix code. These reasons for use, particularly help with understanding and learning, match the benefits discussed in the literature [1], [6].

Trust in this sample (68.8%) is much higher than in the Stack Overflow 2025 survey, in which 33% trusted AI accuracy and 46% distrusted it [5]. Possible explanations are that the respondents are frequent users, that half are beginners with under a year of experience, and that the sample is small and was reached through personal contacts; none of these explanations was tested. The finding that correctness is rated higher than safety (4.25 versus 3.63), while 37.5% have met security issues, is consistent in direction with the weaknesses reported in controlled studies [2], [4], although this survey did not measure the code itself.

The Verification Gap of 12.5 points shows that trust is somewhat higher than combined checking and testing. This agrees with the concern in [3] that confidence can run ahead of care, but the present data show only a modest gap and a majority of calibrated users. The weak association between trust and checking ($\rho = 0.39$), compared with the strong association between trust and perceived safety (ρ

= 0.76), suggests that these respondents trust AI-generated code mainly because they judge it safe and correct, and not because they have verified it. Two cautions apply. All ratings are self-reported and may be overstated, and the cut-off of 4 or 5 is a choice that could move respondents between groups. The measures are descriptive aids and not validated scales.

6 Findings and recommendations

6.1 Findings

- AI use for coding is frequent: 68.8% use it daily, and all respondents have used ChatGPT.
- Perceived correctness (mean 4.25) is higher than perceived safety (mean 3.63), and 37.5% have met security issues.
- Only 62.5% check and 56.3% test AI-generated code before use; the Verification Gap is 12.5 percentage points.
- Half of the respondents are calibrated, 18.8% are at-risk and 25.0% are disengaged.
- Errors or bugs (56.3%) are the most common problem, and only 43.8% report no major problems.

6.2 Recommendations

The recommendations below are suggested by the findings and were not tested directly.

- **Users:** run and test every generated function, including unusual inputs, and apply a security or static-analysis tool before relying on the code.
- **Teachers:** ask for test cases and a short explanation of generated code, and teach common weaknesses such as injection and unsafe input handling.
- **Institutions:** provide guidance on acceptable AI use, disclosure of AI-written code and secure coding.
- **All users:** avoid using AI in place of understanding; code that cannot be explained should not be submitted.

7 Limitations of the study

- The sample is very small (N = 16) and was drawn by convenience sampling, so the results cannot be generalised, and each respondent shifts a percentage by 6.3 points.
- The sample mixes students and professionals, half of the respondents have under a year of experience, and three responses came from the authors.
- All data are self-reported and may be affected by social-desirability bias.
- Only descriptive statistics were used; group differences were not tested for significance.
- The study measures perceptions, not the actual correctness or security of the code used.
- AI tools change quickly, so the findings reflect the period of data collection.

8 Future scope

Future work could use a larger and more varied sample; test associations (for example, between experience and checking) with the chi-square test; combine the survey with a code experiment in

which generated programs are tested with JUnit and scanned with security tools; validate the Verification Gap against measured code quality; and follow the same respondents over time.

9 Conclusion

This survey examined how students and working professionals trust and check AI-generated code. Respondents use AI often and trust it fairly highly, rating correctness above safety, and their trust is somewhat higher than their checking: the Verification Gap is 12.5 percentage points and 18.8% are at-risk users. The proposed Verification Gap and Trust-Verification Matrix offer a simple way to see whether trust is matched by checking. Given the very small convenience sample and the self-reported data, the results are indicative only, but they point to the value of teaching testing and security checking alongside AI-assisted programming.

Declarations

Funding: This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Conflict of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data and Ethics: Participation was voluntary; names and email addresses were collected for validation only and are not reported. Formal ethics review was not required under institutional policy for an anonymous, voluntary survey of adults. Respondents gave informed consent by choosing to submit the form, and a consent statement was shown at the start of the questionnaire.

Data Availability: The data are not publicly available because they contain personal information.

Use of Generative AI: The authors used the generative AI tools Claude [9] and ChatGPT [10] to assist with planning the questionnaire, structuring the manuscript, language editing and formatting. The study design, data collection, analysis and interpretation of the findings are the work of the authors. All reported results are based on genuine survey responses from real respondents; no synthetic or simulated data were used. The authors reviewed and edited all AI-assisted content and take full responsibility for the final manuscript.

Appendix: Questionnaire

A. Profile

Q1. Age (below 18 / 21–25 / 26–30 / 31–40 / above 40)

Q2. What best describes you? (student / software developer / software engineer / other profession)

Q3. How much programming experience do you have? (less than 1 year / 1–2 years / 3–5 years / more than 5 years)

B. Use of AI

Q4. Have you used AI tools for programming or coding? (yes / no)

Q5. Which AI tools have you used? (ChatGPT / GitHub Copilot / Google Gemini / Claude / Microsoft Copilot / other)

Q6. How often do you use AI for coding? (daily / several times a week / sometimes / rarely)

Q7. What do you mainly use AI for? (writing code / fixing errors / understanding code / learning / SQL / testing)

C–E. Ratings, 1 (strongly disagree) to 5 (strongly agree)

Q8. AI-generated code usually gives me the correct solution.

Q9. AI-generated code usually works as expected.

Q10. I feel that AI-generated code is safe to use.

Q11. AI-generated code usually meets my requirements.

Q12. I check AI-generated code before using it.

Q13. I test AI-generated code before using it in my projects.

Q14. I can easily understand and modify AI-generated code.

Q15. I feel confident using AI-generated code after checking it.

Q16. My previous experience with AI affects how much I trust its code.

Q17. I would use AI-generated code in a real project after checking it.

F. Problems and trust

Q18. What problems have you faced with AI-generated code? (wrong logic / wrong output / errors or bugs / incomplete code / security issues / inefficient code / did not meet requirements / none)

Q19. How much do you trust AI-generated code? (1–5)

References

[1] Chen M, et al. Evaluating large language models trained on code. *arXiv* [Preprint]. 2021. arXiv:2107.03374.

[2] Pearce H, Ahmad B, Tan B, Dolan-Gavitt B, Karri R. Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. In: *2022 IEEE Symposium on Security and Privacy (SP)*; 2022. Also published in: Communications of the ACM. 2025;68(2):96–105.

[3] Perry N, Srivastava M, Kumar D, Boneh D. Do users write more insecure code with AI assistants? In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS); 2023. *arXiv*:2211.03622.

[4] Veracode. 2025 GenAI code security report [Internet]. Burlington, MA: Veracode; 2025 Jul 30 [cited 2026 Aug 18]. Available from: https://www.veracode.com

[5] Stack Overflow. 2025 Developer Survey: AI [Internet]. 2025. Available from: https://survey.stackoverflow.co/2025/ai

[6] Kasneci E, et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*. 2023;103:102274. https://doi.org/10.1016/j.lindif.2023.102274

[7] Ji Z, et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*. 2023;55(12):248:1–38. https://doi.org/10.1145/3571730

[8] Creswell JW, Creswell JD. *Research design: qualitative, quantitative, and mixed methods approaches*. 5th ed. Thousand Oaks, CA: SAGE; 2018.

[9] Anthropic. *Claude* [large language model; Internet]. 2026 [cited 2026 Nov 6]. Available from: https://claude.ai

[10] OpenAI. *ChatGPT* [large language model; Internet]. 2026. Available from: https://chatgpt.com