KFUPM
Journal of Undergraduate Research International

Diamond Open Access 52

Article Computer Science

Exploring the Usability of AI-assisted Programming Tools for Non-programmers

Elyas Alamri and Omar Hammad*

Cite https://doi.org/10.64589/juri/231617

Submitted: April 18, 2026 **Revised:** May 21, 2026 **Accepted:** July 09, 2026

ABSTRACT

The increased accessibility of Artificial Intelligence (AI) tools has created debate regarding how well non-experts utilize these tools to complete software tasks and how this may affect the software development labor market. This study investigates the ability of non-expert students to generate functional solutions to Python problems of varying difficulty levels with unrestricted access to generative AI tools. An exploratory user study was conducted with university participants using 15 selected problems from a public programming question database, categorized into easy, medium, and hard difficulty levels. Participants were free to use any AI tools and prompting strategies they preferred. The responses of 11 participants were analyzed. The results showed high success rates for problems of medium difficulty or lower and substantial success rates for hard problems. This indicates that performance declined with increasing difficulty, while remaining relatively high overall. Compared with previous studies, the success rates observed for hard problems were substantially higher, suggesting potential improvements in AI capabilities. The regression analysis suggested that age had a moderate relationship with performance, whereas AI usage frequency and English proficiency showed weak relationships. Despite the exploratory nature and small sample size of this study, the findings suggest that non-experts may effectively leverage modern AI tools to solve programming tasks, although the performance varied by problem type and difficulty level.

Keywords: artificial intelligence, non-programmers, user study, automated code generation, code analysis

1. INTRODUCTION

The rapid sophistication of Artificial Intelligence (AI) tools in recent years has led to commercial AI tools, such as OpenAI's ChatGPT, becoming relatively commonplace among professionals[1], students, and non-experts[2]. The abundance of accessible AI tools has created debate regarding how well non-experts can utilize new Large Language Model (LLM)-based tools to successfully complete software tasks and how this may affect the software development labor market[3].

Several studies have compared AI-generated and human-written code for a specific task to evaluate the effectiveness of AI in achieving a particular programming goal, typically focusing on expert or author use of AI[4]. However, owing to the continuous improvements in AI, it is plausible that these studies are obsolete and paint an inaccurate picture of the current capabilities of AI[5]. Moreover, the existing literature primarily focuses on AI-assisted developers rather than on the capabilities of non-experts using AI[6]. For example, AI tools, such as ChatGPT, have generated outputs equal to or superior to those of university students in introductory programming courses, even when using old AI models[7]. However, few studies have examined the impact of AI access on student performance.

AI tools may be competent at generating code when given a specific task and restrictions on what to use or avoid; however, it is unclear whether this extends to users who do not have substantial programming experience. Studies that have analyzed non-experts include Nguyen et al.[8], who conducted a study on 120 university-level novice programmers. They were asked to write prompts to solve a series of eight problems based on concepts that they had studied in a previous course. They found that only 57% of the problems were eventually solved, concluding that even with LLM assistance, non-experts struggled with programming because of the difficulty of finding an adequate language for prompting. Feldman and Anderson[9] analyzed 67 undergraduates with no prior programming experience who were tasked with solving four specialized Python problems, finding low success rates (an average of 1.4 out of four problems) while using AI. This contrasts with the study by Kazemitabaar et al.,[10] who analyzed the performance of non-expert programmers on Python tasks with and without AI assistance. They found that non-experts were generally able to use AI code generators to successfully solve tasks, while noting that approximately one-third of all non-experts used the task description as the prompt without modification.

Overall, the number of studies analyzing the efficacy of LLM use by non-experts is limited, and the literature presents inconsistent findings. Most studies either use prompts designed by specialist authors or rely on outdated models. Therefore, the aim of this study is to examine the efficacy of non-expert programmers

Published under an exclusive license by KFUPM

using the latest AI models to solve commonly used programming benchmark problems without expert intervention. The remainder of this paper is structured as follows: Section 2 discusses participant recruitment, selection, and the procedure; Section 3 presents the results of the paper and their significance; Conclusions are drawn in Section 4 of the paper.

2. METHODOLOGY

2.1. Participant Recruitment and Screening. 34 university participants were recruited using flyers and links posted on public channels associated with universities. Eligible participants had to be of university age, report their majors and university level, be informed of the nature of the study, and consent to participate before proceeding, Institutional Review Board (IRB) approval was granted for the study. Non-experts were defined as university-aged participants without advanced programming backgrounds, advanced formal computer science training, or competitive programming experience beyond introductory exposure. The screening parameters excluded users who were deemed too advanced for a study analyzing non-experts. This was determined using a pre-study questionnaire asking about previous programming experience, such as taking a mandatory programming course, learning programming languages, and participating in programming competitions. The questionnaire was intended to exclude experts, such as Computer Science graduates. Further filtering excluded participants who failed to solve screening-level questions (labelled as easy) because previous studies have shown that AI has a very high success rate on such questions[11]. It was considered that participants failing to solve easy questions would skew the results; therefore, they were not analyzed further.

After applying these criteria, high-quality participants ($n = 11$) were selected for analysis. Additional data collected from the questionnaire included self-reported English proficiency and AI-usage frequency on a five-point scale.

2.2. Materials and Procedure. The user study involved collecting responses through an online form. Participants were asked to provide preliminary demographic information and consent. After consenting, they completed a form comprising three sections, each consisting of five problems ($n = 15$). Building on the methodology used in previous studies, questions were selected from public programming question databases, specifically LeetCode, to directly compare the success rates across studies. Five easy questions were listed first as a control, followed by five medium-difficulty questions and five hard-difficulty questions. Subsequently, the participants identified the AI tool they had used and submitted the online form.

Problem difficulty labels (easy, medium, and hard) were sourced based on classifications assigned by LeetCode, with the additional criterion that the problem had been created within the last 365 days. This was applied to reduce the likelihood of overfitting resulting from AI training, as noted in previous studies[12]. The questions were selected randomly from the LeetCode database after applying these criteria. Participants were informed, before consenting to the study and answering the questions, that no previous programming knowledge was required, that AI use was allowed, and that they were expected to solve the questions using AI in any manner they deemed appropriate. Participants' AI use was not hindered in any manner, and they were allowed the unrestricted use of AI tools to answer the questions posed.

The participants' responses were manually analyzed and filtered. The accuracy of the solutions provided by students was determined by inserting their code into the LeetCode solution field. Responses were exported from Google Forms to a Microsoft Excel spreadsheet for analysis. Regression and other data analyses were performed by exporting the spreadsheet data to Microsoft Excel for cleaning, and then exporting it to Python for analysis. The data collection and analysis stages are illustrated in Figure 1.

3. RESULTS AND DISCUSSION

The study was designed by integrating elements from previous methodologies. Liu et al.[11] selected problems from the LeetCode database and evaluated AI-generated solutions on success metrics, whereas Feldman and Anderson[9] and Nguyen et al.[8] conducted user studies with non-experts to assess their ability to solve programming tasks using AI.

Building on the work of Liu et al.,[11] who reported success rates of nearly 90% for LLM-generated solutions to LeetCode questions classified as easy, only participants who solved these easy questions were included in the analysis. As previous studies have reported extremely high AI success rates on easy questions, excluding participants who failed these easy tasks were unlikely to substantially bias the sample toward high-performing

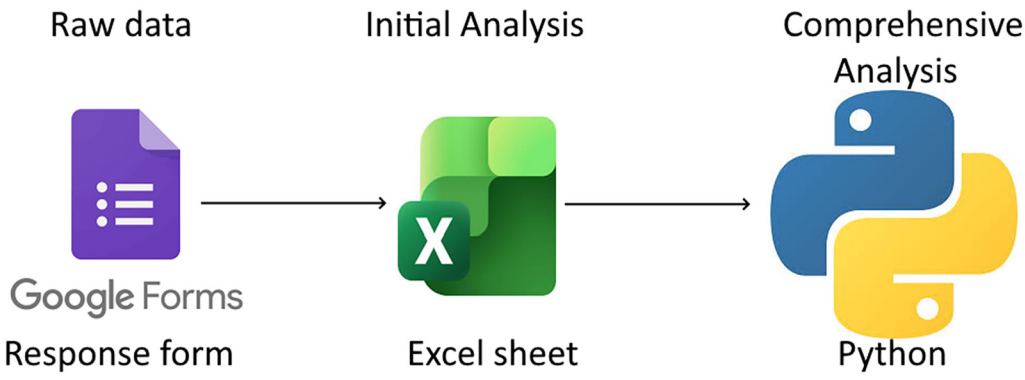

Figure 1. Stages of data collection and analysis

Table 1. *Basic results*

	Easy	Medium	Hard
Minimum (out of five)	5	3	1
Maximum (out of five)	5	4	4
Success rate (%)	100	72.8	60
Mean	5	3.64	3.0
Median	5	4	3
Total Standard deviation (Medium + Hard)		1.29	

users, as such failures were more likely to reflect misunderstandings of instructions or lack of engagement rather than superior programming ability.

The sample mean success score for easy questions was 5/5, indicating a success rate of 100%. The mean success rate for medium questions was lower (3.64/5), corresponding to a success rate of 72.8% (minimum = 3, maximum = 4). The mean success rate for the hard questions was even lower (3/5), corresponding to a success rate of 60% (minimum = 1, maximum = 4). The overall success rate for the medium and hard questions was 66.4%, with a total standard deviation of 1.29. The statistical results are listed in Table 1, and the success rates are shown in Figure 2.

Figure 2 demonstrates a significant decrease in participants' success rates as the problem difficulty increases, although the success rate remained above 50% for hard problems. These results are partially consistent with previous studies. For example, Almanasra and Suwais[13] reported that AI success rates decrease with increasing question difficulty, while Liu et al.[11] reported a 67.4% success rate for a dataset of medium-difficulty LeetCode questions. This study observed success rates for medium-difficulty questions similar to those reported in other studies. However, the success rates for hard-difficulty questions were significantly higher than those of previous studies, as Liu et al.[11] reported success rates of 40% for hard questions, suggesting that AI models may have improved in their ability to solve difficult LeetCode questions.

Additionally, the higher success rate must be interpreted in the context of previous studies that found lower success rates for more recent problems, attributing this to model overfitting[12] resulting from AI model training data that included problems from the same database that was used in this study. Most notably, whereas Liu et al.[11] reported very low success rates for hard-difficulty Python problems created one year before their study (approaching 10%), a much higher success rate (60%) was observed in this study. However, the true magnitude of this increase is uncertain owing to the sample size.

Regression analysis was conducted using Python. Participants' age was associated with a partial R^2 of 0.5, indicating a moderate relationship; however, given the limited sample size, these results should be considered exploratory. Additionally, a regression analysis was conducted on AI usage frequency and English proficiency. Both analyses showed weak associations with the outcome (R^2 <0.2). The R^2 value was 0.194, indicating moderate variability.

The modest variability figures are consistent with previous work[14], which constructed distinct models of how non-experts engage with AI when tasked with solving programming questions. For example, previous studies have shown that while a large percentage of users copy and paste questions, others engage with AI in a more constructive manner. The weak correlations of AI usage frequency and English proficiency with the total success

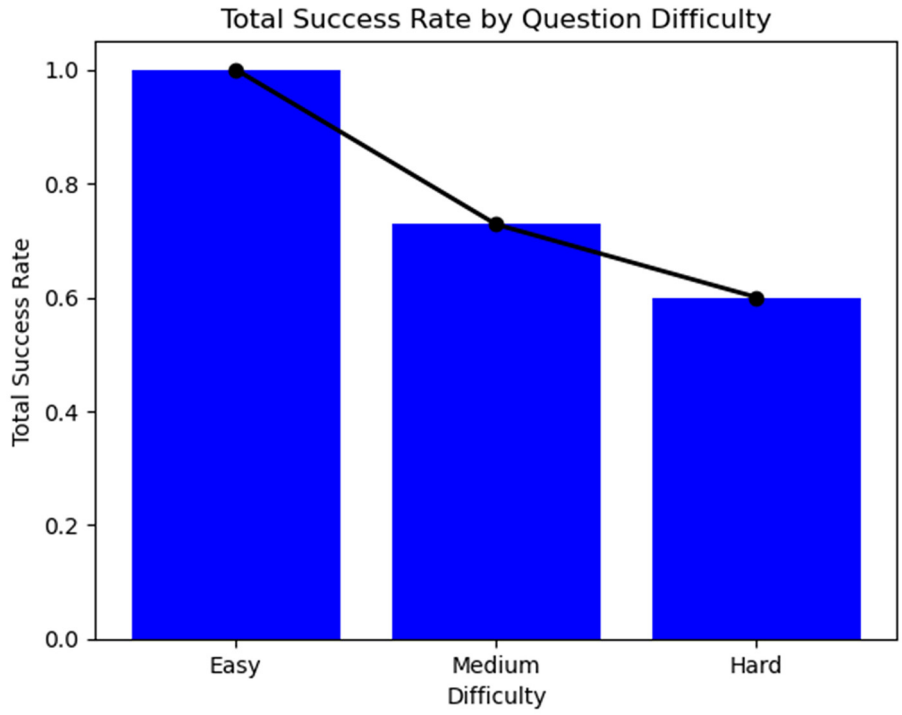

Figure 2. Success rate by question difficulty

rate suggests that AI tools may play a role in equalizing the performance of non-experts with limited English language skills.

Previous studies, such as that of Feldman and Anderson[9], reported a low output rate, resulting in both runtime and syntax errors. No outputs that resulted in errors were identified in the sample in this study, which may indicate improved error-avoidance capabilities in modern AI models. However, further studies with larger sample sizes are required to confirm these observations.

An exception to the general trend was observed for Q5 in the medium-difficulty problem set, which yielded a 0% success rate among participants despite having a substantially higher solution rate on LeetCode (approaching 50%). Notably, this question involved matrix-based operations in addition to standard array manipulation. Prior work has identified array-based tasks as relatively well-handled by LLMs[9]; therefore, the lack of success in this case may point to the increased difficulty associated with matrix-oriented problem structures when approached by non-expert users leveraging AI. Although this observation is based on a single item and cannot be generalized in isolation, the magnitude of the deviation from both the expected performance and the observed trends within this study suggests that this category of problems may warrant more focused investigation in future research.

4. CONCLUSIONS

In conclusion, modern LLM-based AI tools demonstrate a strong capability to assist non-expert users in solving programming problems across varying levels of difficulty, with consistently high success rates across medium-difficulty tasks and lower success rates for hard-difficulty tasks compared to medium-difficulty tasks. The results indicate a decline in performance as the problem difficulty increases, but the overall effectiveness remains substantial, suggesting meaningful improvements in AI capability compared with earlier studies. Although certain problem types, such as matrix-based questions, may still present challenges, the absence of error-prone outputs and relatively high success rates point toward increasing the reliability of these tools.

Despite these findings, this study has several limitations. The relatively small sample size ($n = 11$) limited the generalizability of the results and reduced the statistical strength of the observed relationships, particularly in the regression analysis. Due to the limited sample size, the findings may not fully represent a broader population of non-expert users. Small sample sizes also increased the possibility that individual participant behaviors influence the overall results. Additionally, AI usage was not standardized, and the participants were free to choose any AI tool and interact with it in any manner. This was intended to simulate how real non-experts would interact with programming challenges and to avoid limiting the findings to the behavior of a single AI model.

However, different AI systems may vary in their capabilities and response quality, introducing a lack of experimental control, which may have influenced the results. Similarly, the participants may have differed in how they prompted or interacted with the AI tool, ranging from directly copying the question to engaging in iterative prompting and refinement, which were not measured. Consequently, it is difficult to determine whether the differences in outcomes were primarily caused by participants' abilities, the specific AI tool used, or the interaction strategy employed.

Finally, the absence of a control group without AI-assistance prevented a direct comparison with unaided performance. While the observed success rates suggest that AI tools were beneficial, and the participation criteria aimed at excluding experts, this study could not determine the precise amount that AI contributed to performance owing to the lack of a control group. These factors suggest that the findings should be interpreted as exploratory with appropriate caution. Further research with larger samples and a more detailed analysis of user interaction patterns is required to better understand the mechanisms underlying successful AI-assisted problem-solving.

AFFILIATIONS AND AUTHOR DETAILS

Undergraduate Author

Elyas Alamri – *Department of Information and Computer Science, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia*; 0009-0004-9548-3419
Email: s202337090@kfupm.edu.sa

Corresponding Author

Omar Hammad – *Research Mentor, Professor, Department of Information and Computer Science, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia*; 0009-0009-5581-2843
Email: omarjh@kfupm.edu.sa

ACKNOWLEDGEMENTS

Grammarly and ChatGPT were used for grammar, spelling, and sentence structure corrections. The authors would like to acknowledge the support provided by the Department of Information and Computer Science and the Interdisciplinary Research Center at King Fahd University of Petroleum and Minerals. This research project received no specific funding. The authors have no conflicts of interest to declare.

REFERENCES

(1) Crane, L., Green, M. & Soto, P. Measuring AI uptake in the workplace. FEDS Notes (2025). doi:10.17016/2380-7172.3724

(2) von Garrel, J. & Mayer, J. Artificial intelligence in studies—use of ChatGPT and AI-based tools among students in Germany. Humanit. Soc. Sci. Commun. 10, 799 (2023). doi:10.1057/s41599-023-02304-7

(3) Wang, X., Chen, M. & Chen, N. How artificial intelligence affects the labour force employment structure from the perspective of industrial structure optimisation. Heliyon 10, e26686 (2024). doi:10.1016/j.heliyon.2024.e26686

(4) Oh, S., Lee, K., Park, S., Kim, D. & Kim, H. Poisoned ChatGPT finds work for idle hands: Exploring developers' coding practices with insecure suggestions from poisoned AI models. in IEEE Symposium on Security and Privacy (SP) 1141–1159 (2024). doi:10.1109/SP54263.2024.00046

(5) Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E. A., Prather, J. & Becker, B. A. My AI wants to know if this will be on the

exam: Testing OpenAI's Codex on CS2 programming exercises. in Australasian Computing Education Conference (ACE '23) 97–104 (2023). doi:10.1145/3576123.3576134

(6) Sergeyuk, A. et al. Human-AI Experience in Integrated Development Environments: A Systematic Literature Review. arXiv preprint arXiv:2503.06195 (2025).

(7) Moussiades, L., Zografos, G. & Papakostas, G. GPT-4 vs. GPT-3.5 as coding assistants. Research Square (2024). doi:10.21203/rs.3.rs-3920214/v1

(8) Nguyen, S., Babe, H. M., Zi, Y., Guha, A., Anderson, C. J. & Feldman, M. Q. How beginning programmers and code LLMs (mis)read each other. in CHI Conference on Human Factors in Computing Systems (CHI '24), Article 651 (2024). doi:10.1145/3613904.3642706

(9) Feldman, M. Q. & Anderson, C. J. Non-expert programmers in the generative AI future. in Symposium on Human-Computer Interaction for Work (CHIWORK '24), Article 15 (2024). doi:10.1145/3663384.3663393

(10) Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D. & Grossman, T. How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. in Koli Calling International Conference on Computing Education Research (2023).

(11) Liu, Y., Le-Cong, T., Widyasari, R., Tantithamthavorn, C., Li, L., Le, X.-B. D. & Lo, D. Refining ChatGPT-generated code: Characterizing and mitigating code quality issues. arXiv (2023). doi:10.48550/arXiv.2307.12596

(12) Dai, J., Lu, J., Feng, Y., Ruan, R., Cheng, M., Tan, H. & Guo, Z. MHPP: Exploring the capabilities and limitations of language models beyond basic code generation. arXiv (2024).

(13) Almanasra, S. & Suwais, K. Analysis of ChatGPT-generated codes across multiple programming languages. IEEE Access 13, 23580–23596 (2025).

(14) Güner, H. & Er, E. AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. Educ. Inf. Technol. 30, 12681–12707 (2025). doi:10.1007/s10639-025-13337-7