Jiang, B. et al. (Eds.) (2025). Proceedings of the 33rd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Efficiency or Understanding? Novice Programmers' Problem-Solving With and Without ChatGPT

May Marie P. TALANDRON-FELIPE
University of Science and Technology of Southern Philippines – Main Campus (Alubijid), Philippines
maymarie.talandron-felipe@ustp.edu.ph

Abstract: Large Language Models (LLMs) such as ChatGPT are increasingly present in programming education, but their educational value depends on how they are integrated into student learning. This paper reports on a baseline experiment involving thirty sophomore computing students at a rural state university in the Philippines, classified as novices based on pre-test performance and randomly assigned to AI-assisted and non-AI groups. The AI-assisted group accessed ChatGPT through the free-tier web interface, which in July 2025 defaulted to the GPT-4o model. Students were given two hours to solve three programming problems, followed by a post-test and a focus group discussion. Screen recordings, rubric-based scoring, and observer notes were analyzed to examine coding behaviors and learning outcomes. As expected, the AI-assisted group achieved higher task completion and produced more structurally advanced code, though this often relied on repeated prompting rather than independent debugging. By contrast, the non-AI group generated less polished solutions but engaged more deeply in problem decomposition and iterative debugging, resulting in significantly greater post-test gains. Rather than presenting this efficiency-understanding trade-off as a new discovery, the study contributes contextualized evidence from an underexplored novice setting and provides empirical grounding for ongoing work that designs scaffolded, pedagogically guided integration of LLMs. The findings highlight that unstructured AI use fosters dependency, whereas structured use has the potential to balance efficiency with meaningful conceptual learning.

Keywords: Large Language Models (LLMs), ChatGPT, novice programmers, programming education, coding behaviors

1. Introduction

Large Language Models (LLMs) such as ChatGPT are increasingly being integrated into education, including programming instruction. These tools provide immediate solutions, generate structured code, and offer explanations that can support novice learners. However, while LLMs may enhance efficiency, they also raise concerns about overreliance and reduced opportunities for independent problem-solving skills that are essential for building foundational programming competence.

At the researchers' university, a Learning Continuity Review Program is conducted annually to measure students' mastery of major courses. Although ungraded, it serves as a diagnostic measure of cumulative learning. Recent results revealed consistently low performance in programming courses, particularly Programming 1 and Programming 2, even after course completion. Interviews further revealed that many students had relied heavily on AI tools such as ChatGPT to complete assignments during hybrid learning. While this reliance enabled them to finish tasks, it appeared to limit opportunities for practice and deeper understanding.

International studies have already established that the efficiency–understanding trade-off is a recurring outcome of LLM use in programming education (e.g., Silva et al., 2024; Sun et al., 2024). Yet there remains limited empirical evidence of how this dynamic unfolds among novice programmers in under-resourced university contexts, where AI use intersects with

other challenges such as gaps in preparation, lack of access to computing resources, and language barriers. This study therefore contributes a contextualized baseline by examining how novices in a rural Philippine state university approach programming problems with and without LLM assistance.

Specifically, it compares coding behaviors, debugging strategies, task completion, post-test performance, and student reflections between two groups: one with access to ChatGPT (GPT-4o, July 2025 free-tier version) and one without. Rather than positioning this trade-off as a novel finding, the study's contribution lies in documenting it within an underexplored novice setting and using the results to inform the next phase of research, which involves designing scaffolded, pedagogically guided integration of LLMs in programming education.

2. Potential of LLMs in Programming Education

Introductory programming has long been recognized as one of the most challenging areas in computing education. Novice programmers often encounter difficulties in problem decomposition, debugging, and integrating programming constructs into complete solutions (Robins et al., 2003; Luxton-Reilly et al., 2018). These challenges contribute to high failure and attrition rates in programming courses worldwide. Against this backdrop, the emergence of generative artificial intelligence (AI) tools, particularly large language models (LLMs) such as ChatGPT, has introduced both opportunities and risks for programming education.

Ahmed et al. (2024) evaluated ChatGPT's viability as a teaching assistant in introductory programming. They found that the tool could clarify concepts and provide step-by-step guidance, but was less effective at addressing nuanced misconceptions or personalized debugging. Similarly, Deriba et al. (2023) emphasized in a mini-review that ChatGPT is most effective when positioned as a complement to, rather than a substitute for, student reasoning.

In higher education settings, students have generally expressed positive perceptions of ChatGPT. Ma, Chen, and Konomi (2024, 2025) reported that learners valued its ability to give immediate feedback and alternative explanations in Python courses. Other studies argue that generative AI can help sustain higher-order thinking and programming logic when integrated thoughtfully into curricula (Nathaniel et al., 2025). According to Park and Kim (2025), code suggestions and explanations improved performance, while Matthews et al. (2025) noted that ChatGPT accelerated practice but required guidance to avoid misuse. Collectively, these studies suggest that LLMs may serve as useful scaffolds for novice programmers if their use is carefully structured.

2.1 Risks and Challenges of Overreliance

Research also points to risks associated with heavy dependence on AI tools. In a scientific computing course, Groothuijsen et al. (2024) observed that students frequently used ChatGPT for debugging, explanations, and optimization. While this supported some aspects of learning, instructors worried about declining code quality and reduced collaboration. Similarly, Silva et al. (2024) highlighted the danger of shallow understanding and academic integrity issues when ChatGPT is used without proper oversight.

Humble et al. (2024) explicitly described ChatGPT's dual role as either a tool for "cheaters" or for "AI-enhanced learners." Their findings suggest that outcomes depend heavily on how students employ the tool and how instructors design its integration. Husain (2024) also examined instructors' perspectives, showing enthusiasm for the potential of ChatGPT but concern about uncritical use. Sun et al. (2024) found that while ChatGPT altered student behaviors and perceptions, it could also reduce opportunities for independent reasoning. Likewise, Xue et al. (2024) reported mixed outcomes in an introductory CS course, where efficiency improved but deeper learning did not consistently occur.

From the student perspective, ChatGPT has been described as a form of "augmented intelligence" that speeds up problem-solving but can also foster dependency (Yilmaz & Yilmaz, 2023). Güner and Er (2025) demonstrated that interaction patterns with ChatGPT varied

depending on instructional design and teacher guidance. Andalibi et al. (2024) similarly found that its effectiveness depended on the balance students maintained between AI assistance and independent reasoning.

2.2 Toward Scaffolding Rather Than Substitution

Synthesizing these studies, it becomes evident that ChatGPT's role in programming education is not inherently positive or negative. When structured as scaffolding, it can provide hints, generate examples, and reduce surface-level barriers such as syntax errors, allowing learners to concentrate on problem-solving (Ahmed et al., 2024; Nathaniel et al., 2025). However, when used as a shortcut for complete solutions, it risks bypassing the reasoning processes that novice programmers most need to develop (Groothuijsen et al., 2024; Humble et al., 2024).

This dual potential reflects the findings of the present study, where AI-assisted students demonstrated greater efficiency but less independent reasoning, while those without AI engaged more deeply despite producing fewer polished solutions. The literature therefore underscores the importance of pedagogical design: rather than banning or fully embracing ChatGPT, educators should integrate it in ways that encourage students to reason through problems while using AI as a supportive scaffold.

3. Methods

3.1 Participants

Thirty (30) sophomore computing students participated in the study. All had completed Programming 1 and Programming 2, which covered topics up to multidimensional arrays but not object-oriented programming. Although exposed to core programming content, a pre-test revealed that many struggled to integrate these concepts, consistent with criteria identifying novices as those who possess fragmented or incomplete programming knowledge (Robins et al., 2003; Luxton-Reilly et al., 2018).

Students were divided at random into two groups of fifteen: one permitted to use ChatGPT (AI-assisted) and the other restricted from any AI support (non-AI). Both groups then completed a 20-item pre-test. Analysis of the scores showed that the AI-assisted group ($M = 9.20$, $SD = 1.93$, $n = 15$) and the non-AI group ($M = 9.40$, $SD = 1.69$, $n = 15$) performed at comparable levels, $t(26) = -0.27$, $p = .79$, indicating no meaningful difference at baseline.

3.2 Tasks and Procedure

Each group was given two hours to solve three novice-appropriate programming problems: (1) a palindrome checker, (2) a program to find the second largest number from a list of integers, and (3) a grade calculator that drops the lowest grade and computes the average with an equivalent letter grade. These problems required integration of loops, conditionals, and arrays, but were designed to be solvable within the allotted time.

The non-AI group solved the problems without AI assistance but were allowed to consult their notes for syntax and keyword references. The AI-assisted group accessed ChatGPT through the free-tier web interface, which in July 2025 defaulted to the GPT-4o model. Students were free to use ChatGPT for generating, revising, or debugging code, and prompts/outputs were logged to ensure reproducibility. Both groups also had access to search engines, though search behavior was recorded separately.

Sessions were conducted in a computer laboratory, where each student's screen was recorded to capture their coding process. Three observers were assigned, one per row of terminals, to note behaviors such as reliance on notes, clarifying questions, or use of AI prompts.

To ensure consistency in assessment, task completion was defined as producing a program that satisfied all requirements: dynamic user input, correct execution across possible

test cases, and logical alignment with the task. Completed programs were further classified into categories (Table 1).

Table 1. Criteria for Categorizing Student Programming Solutions

Category	Definition	Example
Completed and correct (10 points)	Logic and syntax correct; dynamic input; passed all test cases; includes retry/exit option.	Fully functional palindrome checker with user input and exit prompt.
Partially completed (8 points)	Core logic correct but requirements unmet (e.g., some inputs hardcoded).	Grade calculator works but number of grades fixed in code.
Completed with correct logic but syntax errors (6 points)	Solution logic is accurate but syntax mistakes prevent proper execution.	Loop condition correct but program fails due to missing colon/bracket.
Completed but with logical errors (4 points)	Code structurally complete but produces incorrect results due to flawed reasoning.	Second-largest number program always outputs the maximum value.
Incomplete (2 points)	Program contains fragments (e.g., input code) but misses the core logic to solve the problem.	Code only takes input without performing palindrome check.
No solution (0 points)	No relevant code submitted for the problem.	Blank file or trivial print statements unrelated to the task.

After the coding session, students took a fifteen-minute break, followed by a post-test identical to the pre-test. The test served as the primary outcome measure for conceptual learning. Code quality and task completion were analyzed as secondary outcomes, providing insight into efficiency and process but not treated as equivalent to learning gain. Finally, separate focus group discussions (FGDs) were conducted to gather qualitative insights into students' experiences.

4. Results

4.1 Primary and Secondary Outcomes

The study assessed two types of outcomes: (a) conceptual learning gains measured by the post-test (primary outcome), and (b) task completion and code quality (secondary outcomes) during the programming session.

4.1.1 Task Completion and Code Quality (secondary outcome)

Across 45 solutions per group (3 problems × 15 students), the AI group produced 39 completed-and-correct solutions and 6 partially completed; none fell into the error categories. The non-AI group produced 28 completed-and-correct, 6 partially completed, 6 completed with correct logic but syntax errors, and 5 completed but with logical errors (Table 2). Using the definitions specified in Methods, this means the AI group more frequently delivered programs that satisfied all requirements (dynamic user input, correct behavior across test cases, and full flow), whereas the non-AI group showed a wider spread across partial, logical-error, and syntax-error outcomes.

Table 2. *Comparison of Task Completion*

Category	AI-Group	Non-AI Group
Completed and correct	39	28
Partially completed	6	6

Completed with correct logic but syntax errors	0	6
Completed but with logical errors	0	5
Incomplete	0	0
No solution	0	0
TOTAL	45	45

Using the rubric-based scoring scheme (maximum of 30 points per student), the AI-assisted group achieved a higher mean solution quality score (M = 29.20, SD = 0.98, n = 15) compared to the non-AI group (M = 25.60, SD = 4.00, n = 15). An independent-samples t-test assuming unequal variances indicated that this difference was statistically significant, t(16) = 3.26, p = .005 (two-tailed). This result, while expected, confirms that ChatGPT assistance produces more polished and complete outputs during time-limited tasks.

4.1.2 Post-test Performance (primary outcome)

The post-test results showed a notable contrast between the two groups. Students in the AI-assisted condition obtained an average score of 10.87 (SD = 1.78, n = 15), while those in the non-AI condition achieved a higher mean of 12.40 (SD = 1.90, n = 15). A comparison of means confirmed that this gap was statistically reliable, t(28) = –2.13, p = .04. Although both groups improved from their pre-test scores, the greater gains of the non-AI group indicate that working without ChatGPT encouraged deeper engagement with programming logic and syntax, even if their solutions were less efficient.

4.2 Coding Structure and Process

Analysis of student submissions and screen recordings revealed distinct differences in coding structure. The AI-assisted group's programs show outputs that were structured, modular, and neatly formatted with functions and consistent naming conventions. In many cases, this produced code that looked more advanced than typical novice work. However, this structure was often adopted directly from ChatGPT, with students showing limited evidence of independent decomposition or debugging. Their process was characterized by copying problem statements into ChatGPT, pasting generated code into the IDE, and returning to the tool for revisions when outputs failed to meet requirements such as dynamic input handling. Although students in this group also had access to a search engine, only three used it, and their searches focused on locating completed solutions rather than consulting syntax references or conceptual explanations. This suggests that both ChatGPT and traditional search were treated as sources of ready-made answers rather than tools for incremental reasoning.

By contrast, the non-AI group produced novice-like codes that can be described as linear, sequential, and less polished, with little to no modularization. Students often began incrementally, first writing preliminary code such as import statements or input-handling snippets, testing these, and only then extending their logic step by step. This bottom-up approach reflected a slower but more effortful form of problem decomposition. Students also asked clarifying questions about problem constraints before coding, demonstrating active engagement in understanding the requirements. Table 3 summarizes the observed coding structure characteristics of the two groups.

Table 3. *Comparison of Coding Structures Between AI-Assisted and Non-AI Groups*

Aspect	AI-Assisted Group	Non-AI Group
Code organization	Structured, modular, use of functions	Linear, sequential, single-block style
Naming conventions	Consistent, descriptive (from LLM outputs)	Inconsistent, often generic (e.g., x, y)
Problem decomposition	Outsourced to ChatGPT (functions auto-generated)	Incremental, bottom-up decomposition

Debugging approach	Relied on re-prompting ChatGPT for revisions	Iterative testing and manual debugging
Clarity of requirements	Few clarifications asked; assumed AI handles interpretation	Asked clarifying questions before coding

4.3 Focus-Group Insights

The focus group discussions further clarified these patterns. Students in the AI-assisted group admitted that they expected ChatGPT to interpret the problems for them and provide ready-made solutions. While they attempted to understand the generated code, they reported difficulty debugging or revising independently, often relying on ChatGPT to make corrections. Several highlighted the additional challenge of expressing what they want or formulating precise prompts in English, noting that vague or incomplete prompts frequently led to outputs that did not satisfy requirements.

Non-AI group students described their primary challenges as constructing the logic of the solutions, recalling relevant concepts, and remembering syntax. Yet they emphasized that working without AI forced them to carefully think through the problems, review their notes, and practice decomposition and debugging. Some contrasted this experience with their freshman year, when reliance on AI for homework made tasks easier but limited their learning. In this activity, though harder, they felt that solving problems without AI reinforced their understanding and confidence.

5. Discussion

The findings of this study reveal a nuanced picture of how novice programmers interact with AI assistance during problem solving. On one hand, the AI-assisted group produced significantly higher quality solutions during the activity, as shown by their rubric-based scores ($M = 29.20$, $SD = 0.98$) compared to the non-AI group ($M = 25.60$, $SD = 4.00$), $t(16) = 3.26$, $p = .005$. Their programs exhibited structured and modular qualities typical of ChatGPT-generated outputs, including consistent formatting and the use of functions. This efficiency advantage translated into more problems being solved correctly within the two-hour timeframe. On the other hand, the post-test results told a different story: the non-AI group outperformed the AI group ($M = 12.40$, $SD = 1.90$ vs. $M = 10.87$, $SD = 1.78$), $t(28) = –2.13$, $p = .04$, indicating stronger conceptual learning when students worked without AI support.

This contrast highlights a trade-off between short-term efficiency and deeper cognitive engagement. The AI group, by relying heavily on ChatGPT, often bypassed the processes of problem decomposition and debugging that are central to novice learning. Screen recordings showed that many students simply pasted entire problem statements into ChatGPT and relied on iterative prompting rather than reasoning through errors. Even when search engines were available, only three students used them, and primarily to locate complete solutions rather than to check references or syntax. These behaviors echo concerns raised in prior research about the risks of overreliance on AI tools, which can compromise authentic engagement with programming concepts (Groothuijsen et al., 2024; Humble et al., 2024; Silva et al., 2024).

By contrast, students in the non-AI group approached problems more incrementally and effortfully. They often began with small code fragments such as input handling, tested these, and then gradually built up the solution. They also asked clarifying questions about problem constraints, which demonstrated active engagement in problem understanding. Although this group produced fewer fully correct solutions within the limited time, their post-test performance suggests that the struggle itself facilitated better consolidation of concepts. This finding resonates with Yilmaz and Yilmaz (2023), who found that students recognize the value of AI as “augmented intelligence” but also acknowledge its potential to create dependency. Similarly, Güner and Er (2025) observed that student-AI interactions vary widely depending on instructional design, underscoring the importance of context in shaping outcomes.

At the same time, the results should not be read as an argument against the use of LLMs in programming education. While unrestricted use appears to encourage overreliance,

prior studies demonstrate that AI tools can be highly effective when employed as scaffolding. Ahmed et al. (2024) showed that ChatGPT can function as a teaching assistant by providing stepwise explanations and supporting novice learners, while Nathaniel et al. (2025) argued that generative AI can sustain higher-order thinking when aligned with problem-solving goals. In this light, the present study suggests that the problem is not the presence of AI but rather how it is integrated into learning environments. Without structure, students may defer too much cognitive work to the tool. With structure, however, AI can be used to guide, hint, or model solutions in ways that support rather than replace student reasoning.

6. Conclusion and Ongoing Work

The findings of this study provide a nuanced understanding of how novice programmers engage with AI assistance during problem solving. The results show two contrasting outcomes. On one hand, the AI-assisted group produced significantly higher-quality solutions during the activity ($M = 29.20$, $SD = 0.98$ vs. $M = 25.60$, $SD = 4.00$, $t(16) = 3.26$, $p = .005$). Their programs displayed structured and modular qualities typical of ChatGPT-generated outputs, including consistent formatting and the use of functions. This efficiency advantage enabled them to complete more tasks within the two-hour timeframe. Such a result, however, was anticipated given prior demonstrations of ChatGPT's ability to generate polished code (Matthews et al., 2025).

More importantly, the post-test outcomes, treated as the primary measure of conceptual learning, revealed that the non-AI group outperformed the AI group ($M = 12.40$, $SD = 1.90$ vs. $M = 10.87$, $SD = 1.78$, $t(28) = –2.13$, $p = .04$). This suggests that students who solved problems independently engaged more deeply with problem decomposition and debugging, leading to greater consolidation of concepts. In other words, while the AI group benefitted from efficiency in the short term, the non-AI group gained longer-term conceptual understanding. Screen recordings showed that AI-assisted students often pasted entire problem statements into ChatGPT, adopted generated code with minimal adaptation, and relied on repeated prompting rather than reasoning through errors. Search engine use was rare and primarily oriented toward locating complete solutions. These behaviors echo concerns raised in earlier studies that unstructured reliance on AI can displace the very reasoning processes novices need to develop (Groothuijsen et al., 2024; Humble et al., 2024; Silva et al., 2024). By contrast, non-AI students approached problems incrementally, constructing code fragments step by step, testing iteratively, and clarifying problem constraints before coding. Although their code was less polished, this struggle appeared to reinforce conceptual learning. At the same time, these findings should not be interpreted as an argument against the use of LLMs in programming education. Prior work has shown that AI tools can be highly effective when integrated as scaffolding rather than as a wholesale replacement for student reasoning (Ahmed et al., 2024; Nathaniel et al., 2025). The present study contributes by situating the efficiency-understanding trade-off in an underexplored context on novice programmers in a rural state university, and by using it as a baseline for designing solution-oriented interventions.

Several actionable strategies emerge for educators:

- Constrain the role of ChatGPT to scaffolding tasks such as offering hints, clarifying syntax, or suggesting debugging strategies, rather than permitting full code generation.
- Require reflective engagement by having students explain, annotate, or modify AI-generated outputs before submission.
- Embed AI use in structured activities (e.g., guided labs, pair programming with AI) where reasoning steps remain visible and accountable.
- Encourage critical comparison by asking students to evaluate AI-generated solutions against their own attempts.

Taken together, these practices align with the principle that the educational value of AI is not in its availability but in how instructors design its use. Without structure, students may defer too much cognitive work to the tool; with structure, AI can function as a productive scaffold that balances efficiency with conceptual growth.

Acknowledgements

The author gratefully acknowledges the student participants who voluntarily took part in this study and shared their time, efforts, and insights, which made this research possible.

References

Ahmed, Z., Shanto, S. S., & Jony, A. I. (2024). Potentiality of generative AI tools in higher education: Evaluating ChatGPT's viability as a teaching assistant for introductory programming courses. *STEM Education*, 4(3), 165-182.

Andalibi, M., Marriott, H., Ozsoy, O. E., Zapata-Rivera, L. F., & Abufardeh, S. (2024, April). Investigating the Effects of ChatGPT on Student Learning in Programming Courses. In 2024 *ASEE PSW Conference*.

Deriba, F. G., Sanusi, I. T., & Sunday, A. O. (2023, November). Enhancing computer programming education using chatgpt-a mini review. *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (pp. 1-2).

Groothuijsen, S., van den Beemt, A., Remmers, J. C., & van Meeuwen, L. W. (2024). AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. *Computers and Education: Artificial Intelligence*, 7, 100290.

Güner, H., & Er, E. (2025). AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Education and Information Technologies*, 1-27.

Humble, N., Boustedt, J., Holmgren, H., Milutinovic, G., Seipel, S., & Östberg, A. S. (2024). Cheaters or AI-enhanced learners: Consequences of ChatGPT for programming education. *Electronic Journal of E-Learning*, 22(2), 16-29.

Husain, A. (2024). Potentials of ChatGPT in computer programming: Insights from programming instructors. *Journal of Information Technology Education: Research*, 23, 002.

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J. and Szabo, C., (2018, July). Introductory programming: a systematic literature review. *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (pp. 55-106).

Ma, B., Chen, L., & Konomi, S. I. (2024, July). Enhancing programming education with ChatGPT: A case study on student perceptions and interactions in a Python course. *International Conference on Artificial Intelligence in Education* (pp. 113-126). Cham: Springer Nature Switzerland.

Ma, B., Chen, L., & Konomi, S. I. (2024). Exploring Student Perception and Interaction Using ChatGPT in Programming Education. *International Association for Development of the Information Society.*

Matthews, R., Ramayah, B., & Yong, S. T. (2025). Merits and Pitfalls of Using ChatGPT in Programming Courses: A Case Study. *The Southeast Asian Conference on Education.*

Nathaniel, J., Oyelere, S. S., Suhonen, J., & Tedre, M. (2025). Investigating the impact of Generative AI integration on the Sustenance of Higher-Order Thinking Skills and understanding of programming logic in Programming Education. *Computers and Education: Artificial Intelligence*, 100460.

Park, A., & Kim, T. (2025). Code suggestions and explanations in programming learning: Use of ChatGPT and performance. *The International Journal of Management Education*, 23(2), 101119.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172.

Silva, C. A. G. D., Ramos, F. N., De Moraes, R. V., & Santos, E. L. D. (2024). ChatGPT: Challenges and benefits in software programming for higher education. *Sustainability*, 16(3), 1245.

Sun, D., Boudouaia, A., Zhu, C., & Li, Y. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. *International Journal of Educational Technology in Higher Education*, 21(1), 14.

Xue, Y., Chen, H., Bai, G. R., Tairas, R., & Huang, Y. (2024, April). Does ChatGPT help with introductory programming? An experiment of students using ChatGPT in CS1. *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training* (pp. 331-341).

Yilmaz, R., & Yilmaz, F. G. K. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. *Computers in Human Behavior: Artificial Humans*, 1(2), 100005.