AI-GENERATED ERRORS AS A LEARNING TOOL: IMPROVING PROGRAMMING EDUCATION THROUGH ERROR CORRECTION

Spartak Sakibayev, Zhetysu University named after Ilyas Zhansugurov
Bela Sakibayeva, Zhetysu University
Darkhan Toybazarov, Zhetysu University named after Ilyas Zhansugurov

ABSTRACT

This study examines the utilization of AI-generated errors in programming education, with a focus on students' ability to identify and correct errors intentionally introduced into AI-generated code. The primary objective is to evaluate the effectiveness of this approach in enhancing students' programming competence and academic performance. Participants engaged with AI-generated code containing predefined errors, and their error detection and correction skills were assessed. The findings indicate that this method supports the development of programming proficiency and contributes to improved academic outcomes. This study adds to the existing literature on AI in education, providing a basis for future research on integrating AI tools into programming instruction.

Keywords*: AI-generated error, AI chatbot, programming instruction, coding club, debugging, error detection and correction*

INTRODUCTION

Integration of artificial intelligence (AI)-based tools has become a defining characteristic of contemporary education. Holmes et al. (2019) assert that AI has become pervasive across various educational domains, influencing instructional processes and practices. According to Daun and Brings (2023) and Raffaghelli et al. (2022), educators regard AI as an essential component of their teaching methodologies. Similarly, research by Albayati (2024), Malik et al. (2021), Sanchez-Ruiz et al. (2023), and Shoufan (2023) indicates that students perceive AI as a valuable tool that enhances their motivation to learn. Furthermore, Chichekian & Benteux (2022), Lim et al. (2023), and Wang et al. (2023) emphasize that advancements in AI technologies continue to shape both current and emerging trends in education.

The influence of AI is particularly evident in computer programming education, as demonstrated by Rahman and Watanobe (2023) and Gezgin et al. (2024). Within this field, AI chatbots have emerged as widely utilized tools (Groothuijsen et al., 2024). Luo et al. (2022) define an AI chatbot as a software application that interacts with users in natural language, thereby enhancing accessibility and engagement.

In programming education, AI chatbots serve various instructional functions. Moon et al. (2023) report that students frequently employ AI chatbots as teaching and mentoring resources that support their general learning needs. Okonkwo and Ade-Ibijola (2021) further highlight that AI chatbots facilitate personalized learning experiences tailored to individual students. However, Groothuijsen et al. (2024) observe that AI chatbots are primarily used for technical tasks, such as debugging, error detection, and optimization. Similar findings are presented by Biswas (2023), Gezgin et al. (2024), and Surameery and Shakor (2023), who indicate

that AI chatbots have become integral tools for assisting students in debugging and resolving errors when solving programming problems.

This integration with AI chatbots aligns closely with the principles of constructivist learning theory, which makes a primary emphasis on active, student-centered engagement with knowledge. According to Kumar Shah (2019), in this theory, learning is most effective when students construct meaning through the processes of problem-solving and reflective thinking—activities that are inherent in such activities as debugging and correcting code.

Additionally, the widespread adoption and educational effectiveness of AI chatbots can be sufficiently examined from the point of view of the Technology Acceptance Model (TAM). In the educational context this model proposes that perceived usefulness and ease of use significantly affect students' acceptance of technology. The proposal is critical to understanding students' willingness to engage with AI tools in learning environments (Legramante et al., 2023).

Despite their utility, AI chatbots pose a risk of generating incorrect information, as noted by Chinonso et al. (2023) and Groothuijsen et al. (2024). Additionally, Groothuijsen et al. (2024) highlight the uncertainty surrounding how students and instructors should approach errors in AI-generated responses to maximize their educational benefit.

Addressing this research gap, the present study examines the potential for developing students' programming competence by engaging them in identifying and correcting errors in AI chatbot-generated responses. The research question guiding this study is: How does resolving errors in AI chatbot-generated code influence students' competence in computer programming? The findings contribute to the broader discourse on AI in education and provide insights for structuring effective learning processes in programming instruction.

METHOD

To investigate the research question, an experiment was conducted within an extracurricular C++ programming club at the Department of Physics and Mathematics at Zhetysu University, Republic of Kazakhstan. The club is attended annually by 40 second-year students who have prior experience with fundamental programming concepts in C or C++. Its primary objective is to provide in-depth instruction on object-oriented design, templates, and template metaprogramming—topics that are typically not covered in sufficient detail during regular coursework due to time constraints. The participating students share a similar academic background, follow the same curriculum, are enrolled in comparable disciplines, and have aligned class schedules.

Prior to the intervention, baseline data were collected to establish the initial equivalence of the experimental and control groups across relevant characteristics. This comparison was necessary to ensure that any observed differences in outcomes could be attributed to the experimental conditions rather than pre-existing disparities between the groups.

Admission to the club is contingent upon passing an entrance examination designed to assess programming proficiency. This assessment consists of a combination of programming problems and theoretical questions aimed at evaluating students' foundational knowledge and problem-solving skills. The maximum achievable score on the examination is 200 points, allowing for a comprehensive evaluation of a wide range of competencies.

The average entrance exam scores were 130 for students in the experimental group and 135 for students in the control group. While a slight difference was observed, the scores fall within a narrow range, indicating that both groups demonstrated comparable levels of initial performance and understanding. Statistical analysis confirmed that these differences were not statistically significant, supporting the assumption that the groups were equivalent prior to the intervention. This minimal variation suggests that the participants' initial preparation and knowledge base were nearly identical. Given the comprehensive nature of the entrance examination, which evaluates a broad spectrum of programming concepts and problem-solving skills, it can be concluded with confidence that both groups entered the study with similar levels of proficiency. Consequently, any differences in study outcomes are unlikely to result from disparities in prior knowledge or preparation, ensuring that both groups had a comparable starting point within the context of this research.

The control group participated in the club one year before the experiment. Meetings were held twice weekly, following a structured format. Students were divided into subgroups of 20 participants each. Each session commenced with an instructor-led explanation of a theoretical concept, delivered using an interactive board. Subsequently, students were presented with an assignment consisting of a code fragment generated by an AI chatbot (ChatGPT) displayed on the board. The assignment required students to analyze the code, explain its functionality, and either optimize it or modify it to incorporate additional features. Students were allocated up to 45 minutes to complete the task. In cases where errors were identified in the provided code, the instructor could request an alternative code fragment from the AI chatbot for further examination and refinement.

At the conclusion of each topic within the curriculum, students completed an examination designed to assess their comprehension of the material covered during club sessions. The examination required students to analyze a code fragment generated by the same AI chatbot, optimizing or extending it with specified new functionality. Additionally, the examination included a task requiring the development of a small application utilizing C++ features introduced during the club sessions. Examination scores were assigned based on the rubric outlined in Table 1.

The simplified rubric presented in Table 1 was implemented to standardize the assessment process and ensure clear, focused evaluation criteria. The primary objective was to minimize ambiguity and enhance efficiency in assessing student performance. By emphasizing key aspects of the assigned tasks while eliminating unnecessary complexity, the rubric facilitated a transparent and consistent evaluation framework applicable to all participants.

Additionally, the use of a simplified rubric improved interrater reliability by ensuring that multiple evaluators assigned similar scores to the same work. This consistency was critical for maintaining the validity of the study, as it reduced the influence of individual rater differences and enabled reliable comparisons across participants.

The three-point scale (0–5–10) provided an appropriate level of granularity for the study's objectives. This scale balanced simplicity with the need to effectively differentiate student performance. The three levels captured a range of student achievement, from incomplete work (0 points) to partial completion with noticeable gaps (5 points) and fully complete, high-quality submissions (10 points). This scale was sufficient to distinguish varying levels of understanding and task completion while maintaining a clear and objective evaluation process. Given the nature of the assignments and the study's objectives, this approach allowed for the assessment of students' progress without introducing unnecessary complexity. Moreover, the use of this three-scale rubric has been an established practice in the club's activities since its inception.

The instructional format for the experimental group closely mirrored that of the control group, with one key distinction: The code fragments provided to students, generated by an AI chatbot, contained deliberate errors. Students were

Table 1.
Grading Rubric for Final Examination

Grade	Criteria	Completion	Quality	Effort Level	Remarks
10	Assignment is fully complete, meets all requirements, demonstrates thorough understanding, and is of high quality with no errors or omissions.	Fully Completed	Excellent, no errors or omissions	High effort, exceeds expectations	Work shows clear mastery of topic and is well-organized and thoughtful.
5	Assignment is partially complete, meets some requirements, but contains significant gaps, errors, or lacks depth in key areas.	Partially Completed	Contains some errors or omissions	Moderate effort, meets basic requirements	Work is incomplete but demonstrates some understanding.
0	Assignment is not complete at all, does not meet the minimum requirements, or is missing critical components.	Incomplete	Poor quality or missing content	Minimal to no effort	Does not meet the required criteria for completion.

required to identify and correct these errors as part of their assignments. In the final examination, the AI-generated code fragment also included intentional errors, which students were expected to detect and resolve.

The topics covered during club sessions for both groups included custom generic containers, such as AVL trees, splay trees, red-black trees, and hash maps. Additionally, discussions focused on generic algorithms for processing these data structures. Students completed their programming tasks using Ubuntu-based laptops and online C++ coding platforms. Instruction was delivered by the same educators for both groups: one instructor was responsible for theoretical content, while another supervised practical assignment. The final examinations were also administered by the same instructors to ensure consistency in assessment.

The tasks assigned to participants in both the experimental and control groups were comparable in content and complexity. These tasks were structured into distinct topics, which are listed in Table 2.

For example, when studying Binary Search Trees (BST), a student in the control group may have received a C++ implementation of the *Insert*() method, which is designed to add a new node to a tree intended for moderate-sized datasets. While functionally correct, the implementation is highly inefficient. It employs raw loops instead of recursion or structured approaches and traverses the entire tree for each insertion without early termination. Additionally, the method does not include checks for potential memory allocation failures. Alternatively, students may be provided with a method for deleting a node from the tree. This method fails to account for potential errors and includes unnecessary recursive calls when the specified node is not found. In both cases, the code is inefficient, redundant, and lacks essential error handling. The students' task is to identify these inefficiencies, optimize the logic, improve memory management, and implement more efficient algorithms.

In contrast, students in the experimental group received code containing both syntactic and semantic errors, preventing successful compilation. Examples of such errors include missing brackets or semicolons, uninitialized pointers to left and right child nodes, and incorrect comparison operators leading to improper node placement. Additionally, errors may include incorrect return types or parameter types, as well as improperly handled base cases in recursive functions. For instance, when the tree is empty, the algorithm may fail to initialize the root node correctly. Similarly, if the node to be inserted is found during recursive traversal, the method may not correctly terminate recursion, leading to unnecessary further checks or operations that could be avoided with a correct base case implementation.

The provided code fragments were intentionally kept small, ensuring they could be displayed on a single screen without requiring scrolling. Table 3 presents an example of code provided to students in the experimental group, which includes syntactic errors. This code was used in the study of Binary Search Trees (BST).

Table 2.
Topics of Tasks Assigned in the Club

No	Topic	Data Structure
1	Operations of insertion, lookup, and deletion for a Binary Search Tree (BST).	BST
2	Returning the minimum and maximum values stored in the tree.	BST
3	Rebalancing on insertion and deletion in an AVL tree.	AVL Tree
4	Checking if a given binary tree is an AVL Tree, using height-balance criteria.	AVL Tree
5	Insertion, deletion, and rebalancing for a Red-Black Tree with adherence to color rules.	Red-Black Tree
6	Visualization for Red-Black Trees with highlighting each node's color (red or black).	Red-Black Tree
7	Insert, delete, and search operations for a HashMap with separate chaining for collision handling.	HashMap
8	Dynamic resizing of the HashMap based on load factor to optimize performance.	HashMap
9	Optimization of the performance of BST, AVL Tree, and Red-Black Tree for insertion and search operations.	BST, AVL Tree, Red-Black Tree
10	Choosing the optimal data structure for the given dataset.	All

Table 3.
Example Code with Syntactic Errors for BST Study

```
void insert(TreeNode* node, int val) {
    if (val < node->value)
        if (node->left == nullptr)
            node->left = new
TreeNode(val);
        else
        insert(node->left, val);
```

Table 4 provides an example of code presented to students in the experimental group that contains semantic errors. This code is used in the study of Binary Search Trees (BST) and exhibits flaws in its logical structure or program flow, resulting in incorrect behavior or outcomes despite being syntactically correct.

Table 4.
Example Code with Semantic Errors for BST Study

```
void insertNode(TreeNode* node, int val) {
    if (node == nullptr) {
        node = new TreeNode(val);
        return;
    }
    if (val < node->val) {
        insertNode(node->left, val);
    } else if (val > node->val) {
        insertNode(node->right, val);
    }
}
```

The present study utilized the tasks outlined in Table 2, implemented as problems similar to those presented in Tables 3 and 4, as the primary assessment method to evaluate the effectiveness of the learning approach employed during the club's meetings and to analyze the performance of the participating students. Data analysis was conducted using descriptive statistical methods via an online spreadsheet application and the SPSS software package. Specifically, the analysis focused on calculating participants' average grades based on their performance in assigned tasks. Grades were recorded in the spreadsheet and the SPSS environment, where built-in statistical functions were used to compute the mean score, providing an overall measure of student performance. A mean score of 7.75 or higher was defined as a positive outcome for the experimental group, while the control group achieved a mean score of 6.25.

Following the completion of the experiment and data collection, statistical analysis was conducted to compare the academic performance of the experimental and control groups. A two-sample *t*-test was applied to compare the mean performance scores between the groups. This test was selected because it is appropriate for assessing differences in means between two independent samples.

The *p*-value for the *t*-test was calculated to determine whether the observed differences in academic performance were statistically significant. A significance level (α) of 0.05 was established a priori, with *p*-values below this threshold indicating that the differences between groups were unlikely to have occurred by chance.

In addition to performance-based evaluation, a questionnaire was administered to gather further insights into participants' experiences and perceptions. This qualitative data contextualized the performance metrics, offering a more comprehensive understanding of participants' views on their progress and the effectiveness of the club's activities. The questionnaire data were analyzed using descriptive statistics within the same online spreadsheet application and the SPSS package. The target outcome for the questionnaire was that at least 70% of students would respond "yes" to both the first and second questions. The contents of the questionnaire are summarized in Table 5.

Table 5.
Questionnaire for Experimental Group

Question No.	Question
1	Has your competence in computer programming improved as a result of participating in the club activities? (Yes/No)
2	Has your interest in computer programming increased as a result of participating in the club activities? (Yes/No)
3	Do you find that the tasks and activities from the club could be effectively incorporated into regular lessons? (Yes/No)

The questionnaire was administered to the experimental group immediately following the final examination via email. Two participants did not submit responses.

The distribution and processing of the questionnaire, as well as all other experimental procedures, were conducted with the informed consent of the club's participants. They were fully briefed on the study's objectives, methodology, and the intention to publish the findings in a research paper.

RESULTS

The examination grades of the first subgroup of the experimental group, following the completion of each topic outlined in Table 2, are presented in Table 6.

Table 6.

Performance of the first subgroup of the experimental group

Topic No	Mean Score
1	6.75
2	8.5
3	7.5
4	8.0
5	7.5
6	7.75
7	7.5
8	7.5
9	7.75
10	9.0
Resulting mean score = 7.775	

Meanwhile, the grades obtained by the first subgroup of the control group while working on the same topics are presented in Table 7.

Table 7.

Performance of the First Subgroup of the Control Group

Topic No	Mean Score
1	6.0
2	6.5
3	6.25
4	6.5
5	6.0
6	5.5
7	6.0
8	6.5
9	6.0
10	7.0
Resulting mean score = 6.225	

Table 8 presents the grades achieved by the second subgroup of the experimental group on the topics from Table 2.

Table 8.

Performance of the Second Subgroup of the Experimental Group

Topic No	Mean Score
1	8.5
2	9.25
3	9.5
4	9.25
5	9.25
6	9.5
7	9.0
8	9.5
9	9.25
10	9.5
Resulting mean score = 9.25	

Next, Table 9 presents the grades of the second subgroup of the control group on the same topics.

Table 9.

Performance of the Second Subgroup of the Control Group

Topic No	Mean Score
1	6.5
2	7.5
3	7.5
4	7.5
5	7.5
6	7.5
7	7.0
8	7.25
9	7.0
10	7.25
Resulting mean score = 7.25	

An analysis of the grades presented in the data tables indicates that both the first and second subgroups within the experimental group consistently outperformed their counterparts in the control group. This trend suggests that the interventions applied to the experimental group had a positive impact on academic performance, with measurable improvements observed.

In order to determine whether the observed difference in topic-level performance between two

student subgroups was statistically significant, *t*-test was conducted using the SPSS package. This test was appropriate because both groups were assessed on the same set of 10 topics, allowing for an evaluation of whether the mean scores differed systematically across these matched topic pairs.

The null hypothesis ($H\square$) stated that there was no difference in the mean scores between the two subgroups across the 10 topics. In contrast, the alternative hypothesis ($H\square$) posited that there was a significant difference in the mean scores between the two groups. A rejection of the null hypothesis would suggest that the observed difference in performance was statistically significant. Table 10 contains the paired sample statistics obtained during the *t*-test, involving the first subgroups of the experimental group (EG) and control group (CG).

Table 10.
Paired Sample Statistics of the First Subgroups of Control and Experimental Groups

Pair	*Mean* (EG)	*Std. Dev.* (EG)	*Mean* (CG)	*Std. Dev.* (CG)	N
EG-CG	7.775	0.617	6.225	0.416	10

Following the analysis of the paired sample statistics, the paired samples correlations for the same subgroups were examined to assess the strength and direction of the relationship between the performance scores across the 10 topics. The correlations are given in Table 11.

Table 11.
Paired Samples Correlations for the First Subgroups of Control and Experimental Groups

Pair	Correlation	N
EG-CG	*0.652*	*10*

The correlation (r) between the experimental group and the control group is approximately 0.652, indicating a moderate positive relationship between the two sets of scores.

Building on the paired sample statistics, a paired samples *t*-test was conducted to determine whether there was a statistically significant difference in performance between the subgroups across the 10 topics. The results of the test are displayed in Table 12.

Table 12.
Paired Samples t-test for the First Subgroups of Control and Experimental Groups

Pair	*Mean Difference*	*Std. Dev.*	*t*	df	Sig.	*p*-value
EG-CG	1.55	0.468	10.46	9	.041	0.041

As shown in Table 12 the difference in average grades between the experimental and control groups is statistically significant ($p < 0.05$). These results suggest that the experimental subgroup performed significantly better across all topics.

Table 13 contains the paired sample statistics obtained during the *t*-test, involving the second subgroups of the experimental and control groups.

Table 13.
Paired Sample Statistics of the Second Subgroups of Control and Experimental Groups

Pair	*Mean* (EG)	*Std. Dev.* (EG)	*Mean* (CG)	*Std. Dev.* (CG)	*N*
EG-CG	*9.25*	*0.27*	*7.25*	*0.40*	*10*

Following the analysis of the paired sample statistics, the paired samples correlations for the same subgroups were examined to assess the strength and direction of the relationship between the performance scores across the 10 topics. The correlations are given in Table 14.

Table 14.
Paired Samples Correlations for the Second Subgroups of Control and Experimental Groups

Pair	Correlation	Sig. (2-tailed)	N
EG-CG	~0.90	0.0053	10

The correlation (r) between the experimental group and the control group is approximately 0.90, indicating a strong positive relationship between the two sets of scores.

Building on the paired sample statistics, a paired samples *t*-test was conducted to determine whether there was a statistically significant difference in performance between the subgroups across the 10 topics. The results of the test are displayed in Table 15.

Table 15.
Paired Samples t-test for the First Subgroups of Control and Experimental Groups

Pair	Mean Difference	t-statistics	Df	p-value
EG-CG	**2.00**	**3.77**	9	0.0053

This suggests that there is a highly significant difference between the experimental and control groups. The experimental group performs significantly better than the control group based on the analysis. Thus, with this *p*-value, it is clear that the null hypothesis is rejected, and the evidence strongly supports a significant difference between the groups.

This result strongly supports the conclusion that the interventions had a meaningful impact on the academic outcomes of the experimental group. The low *p*-value provides strong evidence that the observed performance advantage is unlikely to have occurred by chance, reinforcing the effectiveness of the study's interventions.

These findings provide strong evidence that the interventions had a meaningful effect on the academic outcomes of the experimental group. The low *p*-value further supports the conclusion that the observed performance improvements were not due to chance, reinforcing the effectiveness of the applied instructional approach.

The results of the questionnaire administered at the conclusion of the experiment are presented in Table 16.

Table 16.
Questionnaire Results

Question No	Percentage of *Yes* Responses (%)	Proportion	Confidence Intervals (CIs)
1	75	0.75	95% [0.61, 0.89]
2	85	0.85	95% [0.74, 0.96]
3	85	0.85	95% [0.74, 0.96]

Table 16 illustrates that the majority of interviewed students demonstrated a strong preference for the methodology employed within the club. Moreover, a significant number of students expressed a preference for the integration of this methodology into their regular academic curriculum.

As related to confidence intervals (CIs), 75% of students reported an improvement in programming skills (95% CI: [0.61, 0.89]) indicating strong confidence in the intervention's effectiveness. Additionally, a binomial test confirmed that the proportion of students reporting increased programming interest (85%) was significantly higher than a neutral expectation of 50%, $p < .001$, suggesting strong student engagement.

The tasks assigned within the extracurricular programming club provided students with the opportunity to engage with the methodology, generating measurable performance data across key programming competencies. The questionnaire responses captured a range of perspectives on task engagement, perceived learning, and skill application, offering a comprehensive overview of students' experiences with the instructional approach. Collectively, these findings provide a robust data set that serves as the basis for further analysis in the Discussion section.

DISCUSSION

The results of both the questionnaire and the tasks completed within the club indicate that the experiment successfully achieved all predefined quantitative objectives. Data collected from the questionnaire, reflecting participants' perceptions and experiences, alongside performance metrics derived from task completion, demonstrate that the expected outcomes were met. Notably, participant responses indicated a high level of engagement and satisfaction with the methodologies employed, while task results confirmed the effectiveness of the interventions in developing targeted programming competencies. These findings affirm the robustness of the experimental design, as all objectives were fully realized.

The results of this study align with previous research, including the findings of Groothuijsen et al. (2024) and Xie et al. (2019), which emphasize the benefits of integrating AI chatbots into programming education to enhance the learning process. Furthermore, these results corroborate those of Rahman & Watanobe (2023), who identified AI chatbots as valuable tools for addressing programming errors. This study extends Rahman's findings by demonstrating that requiring students to detect

and correct intentional errors in AI-generated code is an effective approach to improving programming proficiency.

Engaging students in analyzing and debugging flawed code encourages deeper interaction with programming concepts, reinforcing their understanding of syntax, logic, and common error patterns. This approach not only enhances technical proficiency but also fosters critical thinking and problem-solving skills, which have broader academic applications. Additionally, exposure to AI-generated errors familiarizes students with real-world debugging challenges, better preparing them to independently identify and resolve similar issues in their own work. By practicing error correction within a structured and supportive learning environment, students may also gain increased confidence in debugging. The use of AI chatbots for deliberate error generation offers a novel means of supplementing traditional programming exercises, providing a controlled setting for students to learn from mistakes and improve their coding accuracy and resilience.

One of the broader implications of the present study is that, unlike traditional learning environments, it enables students to engage with artificial intelligence tools in a more productive and beneficial manner. The present study's approach aligns with constructivist learning theory by motivating students for active and self-directed learning activities, and is further supported by the Technology Acceptance Model, which emphasizes the importance of perceived usefulness and ease of use in technology adoption. Rather than remaining passive recipients of educational information generated by AI chatbots, students are motivated to adopt a more active learning approach. The methodology presented in this paper encourages an attitude where learners actively question, test, and adjust AI-generated responses—thus, transforming the chatbot from a source of static information into a dynamic and interactive learning partner. This active engagement not only deepens students' understanding within the context of programming but also develops broader digital literacy skills essential in today's AI-driven world. When employed in this way, AI chatbots are not merely instructional aids but become valuable companions in the learning process, contributing to a more interactive and motivating educational experience across various academic disciplines, not just computer programming.

Despite these positive outcomes, several limitations of this study should be acknowledged. First, the study was limited to a single programming language, which may restrict the generalizability of the findings to other languages with different syntax and structures. Additionally, only topics related to generic containers were included in the assessment, leaving other fundamental programming concepts unexplored within this methodology. The study also did not examine whether similar results would be obtained if students themselves generated erroneous code using AI chatbots. Furthermore, the participant sample size was relatively small, which may limit the generalizability of the conclusions. Future research could address these limitations by incorporating multiple programming languages, expanding the range of topics covered, and exploring more interactive forms of student-AI engagement.

Educators interested in replicating this approach can implement the method using publicly available AI tools, such as ChatGPT, to generate code fragments with intentional syntax or logic errors. These fragments can be used in 30–45 minute sessions where students work individually or in small groups to identify, understand, and correct the errors. Assessment can be based on students' ability to accurately detect and fix these errors, with the instructor providing constructive feedback where necessary. To ensure accessibility of this method for learners of varying skill levels, instructors may provide hints or guidance when needed. Minimal technical requirements are necessary, and teachers are encouraged to review the AI-generated material in advance and gain familiarity with the tools to ensure the activities align with course objectives.

CONCLUSION

This study underscores the potential of AI chatbots in programming education by enabling students to engage with deliberately introduced errors in code. The findings indicate that this approach effectively reinforces students' understanding of key programming concepts while enhancing their problem-solving abilities. Through direct engagement with debugging tasks, students

develop hands-on experience that contributes to increased confidence and technical proficiency.

While the study provides valuable insights, several limitations must be considered. The exclusive focus on a single programming language and the specific topic of generic containers may limit the broader applicability of the results. Additionally, the study did not examine the effects of allowing students to actively generate erroneous code using the chatbot, an area that warrants further investigation. Future research could address these limitations by incorporating multiple programming languages, expanding the range of covered topics, and exploring alternative student-AI interaction models.

Overall, the study contributes to the growing body of research on AI in education, offering a foundation for further exploration into the role of AI chatbots in programming instruction. By addressing the identified limitations, this methodology has the potential to further enhance programming education across diverse learning contexts.

In addition to the aforementioned areas to be explored further, it is important to explore the potential impact of using other student-AI interaction models as well. The current study primarily focuses on a model that implies that students actively request erroneous code from the chatbot. Future research could deal with the model of passive engagement, where students work with pregenerated errors, or the adaptive model, where AI chatbot tailors the errors aligned with student performance. Testing these models could offer a deeper insight into the methodology discussed in the current study and contribute to more personalized and adaptive learning experiences.

Furthermore, the possible implications of integrating AI chatbots into programming education are not limited to their use in individual learning settings. Advancements in AI-driven educational tools allow them to play a significant role in forming future programming curricula by considering personalized, interactive learning experiences. The integration of AI could also contribute to creating adaptive learning environments, where students receive real-time feedback and guidance to improve their academic performance and learning outcomes.

ACKNOWLEDGEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request. On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Albayati, H. (2024). Investigating undergraduate students' perceptions and awareness of using ChatGPT as a regular assistance tool: A user acceptance perspective study. Computers in Education: Artificial Intelligence, 6, Article 100203. https://doi.org/10.1016/j.caeai.2024.100203

Biswas, S. (2023). Role of ChatGPT in computer programming: ChatGPT in computer programming. Mesopotamian Journal of Computer Science, 2023, 9–15. https://doi.org/10.58496/MJCSC/2023/002

Chichekian, T., & Benteux, B. (2022). The potential of learning with (and not from) artificial intelligence in education. Frontiers in Artificial Intelligence, 5, 903051. https://doi.org/10.3389/frai.2022.903051

Chinonso, O. E., Theresa, A. M. E., & Aduke, T. C. (2023). ChatGPT for teaching, learning and research: Prospects and challenges. Global Academic Journal of Humanities and Social Sciences, 5(2), 33–40. https://doi.org/10.36348/gajhss.2023.v05i02.001

Daun, M., & Brings, J. (2023). How ChatGPT will change software engineering education. In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education, vol. 1 (pp. 110–116). Association for Computing Machinery. https://doi.org/10.1145/3587102.3588815

Gezgin, D. M., Mert, S., Kesici, A. İ., & Yıldırım, S. (2024). Understanding university students' intentions to use chatbots in computer programming education: A quantitative study. Sakarya University Journal of Education, 14(Special Issue-AI in Education), 142–158. https://doi.org/10.19126/suje.1426980

Groothuijsen, S., van den Beemt, A., Remmers, J., & 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. https://doi.org/10.1016/j.caeai.2024.100290

Holmes, W., Bialik, M., & Fadel, C. (2019). Artificial intelligence in education: Promises and implications for teaching and learning. Center for Curriculum Redesign.

Kumar Shah, R. (2019). Effective constructivist teaching learning in the classroom. Shanlax International Journal of Education, 7(4), 1–13. https://doi.org/10.34293/education.v7i4.600

Legramante, D., Azevedo, A., & Azevedo, J. M. (2023). Integration of the technology acceptance model and the information systems success model in the analysis of Moodle's satisfaction and continuity of use. International Journal of Information and Learning Technology, 40(5), 467–484. https://doi.org/10.1108/IJILT-12-2022-0231

Lim, W. M., Gunasekara, A., Pallant, J. L., Pallant, J. I., & Pechenkina, E. (2023). Generative AI and the future of education: Ragnarök or reformation? A paradoxical perspective from management educators. The International Journal of Management Education, 21(2), 100790.

Luo, B., Lau, R. Y., Li, C., & Si, Y. W. (2022). A critical review of state‐of‐the‐art chatbots designs and applications. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 12(1), e1434. https://doi.org/10.1002/widm.1434

Malik, R., Sharma, A., Trivedi, S., & Mishra, R. (2021). Adoption of chatbots for learning among university students: Role of perceived convenience and enhanced performance. International Journal of Emerging Technologies in Learning (iJET), 16(18), 200–212. https://doi.org/10.3991/ijet.v16i18.24315

Moon, J., Yang, R., Cha, S., & Kim, S. B. (2023). ChatGPT vs Mentor: Programming Language Learning Assistance System for Beginners. In IEEE 8th International Conference on Software Engineering and Computer Systems (ICSECS) (pp. 106–110). IEEE. https://doi.org/10.1109/ICSECS58457.2023.10256295

Okonkwo, C. W., & Ade-Ibijola, A. (2021). Evaluating the ethical implications of using chatbot systems in higher education. In U. G. Singh & C. S. Nair (Eds.), digiTAL 2021 Conference Proceedings (pp. 68–77). digiTAL2K. https://icdigital.org.za/wp-content/uploads/2021/12/digiTAL-2021-Conference-Proceedings.pdf

Raffaghelli, J. E., Rodríguez, M. E., Guerrero-Roldán, A. E., & Baneres, D. (2022). Applying the UTAUT model to explain the students' acceptance of an early warning system in Higher Education. Computers & Education, 182, 104468. https://doi.org/10.1016/j.compedu.2022.104468

Rahman, M. M., & Watanobe, Y. (2023). ChatGPT for education and research: Opportunities, threats, and strategies. Applied Sciences, 13(9), 5783. https://doi.org/10.3390/app13095783

Sánchez-Ruiz, L. M., Moll-López, S., Núñez-Pérez, A., Moraño-Fernández, J. A., & Vega-Fleitas, E. (2023). ChatGPT challenges blended learning methodologies in engineering education: A case study in mathematics. Applied Sciences, 13, 6039. https://doi.org/10.3390/app13106039

Shoufan, A. (2023). Exploring students' perceptions of ChatGPT: Thematic analysis and follow-up survey. IEEE Access, 11, 38805–38818. https://doi.org/10.1109/ACCESS.2023.3268224

Surameery, N. M. S., & Shakor, M. Y. (2023). Use chat GPT to solve programming bugs. International Journal of Information Technology & Computer Engineering (IJITC), 3(01), 17–22. https://doi.org/10.55529/ijitc.31.17.22

Wang, T., Lund, B. D., Marengo, A., Pagano, A., Mannuru, N. R., Teel, Z. A., & Pange, J. (2023). Exploring the potential impact of artificial intelligence (AI) on international students in higher

education: Generative AI, chatbots, analytics, and international student success. Applied Sciences, 13(11), 6716. https://doi.org/10.3390/app13116716

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., Tan, A. H., Hwa, L., Li, M., & Ko, A. J. (2019). A theory of instruction for introductory programming skills. Computer Science Education, 29(2-3), 205–253. https://doi.org/10.1080/08993408.2019.1565235