Cogent Education

ISSN: 2331-186X (Online) Journal homepage: www.tandfonline.com/journals/oaed20

Evaluating the impact of Large Language Models on problem-solving skills in programming debugging of IT undergraduates

Fathima Riztha, Ruwan Wickramarachchi, PPG Dinesh Asanka & Mathishi Adya Dissanayake

To cite this article: Fathima Riztha, Ruwan Wickramarachchi, PPG Dinesh Asanka & Mathishi Adya Dissanayake (2026) Evaluating the impact of Large Language Models on problem-solving skills in programming debugging of IT undergraduates, Cogent Education, 13:1, 2658267, DOI: 10.1080/2331186X.2026.2658267

To link to this article: https://doi.org/10.1080/2331186X.2026.2658267

Published online: 27 Apr 2026.

COGENT EDUCATION
2026, VOL. 13, NO. 1, 2658267
https://doi.org/10.1080/2331186X.2026.2658267

INFORMATION & COMMUNICATIONS TECHNOLOGY IN EDUCATION
| RESEARCH ARTICLE

OPEN ACCESS Check for updates

Evaluating the impact of Large Language Models on problem-solving skills in programming debugging of IT undergraduates

Fathima Riztha[a], Ruwan Wickramarachchi[a], PPG Dinesh Asanka[a] and Mathishi Adya Dissanayake[b]

[a]Department of Industrial Management, Faculty of Science, University of Kelaniya, Kelaniya, Sri Lanka; [b]Department of Information Technology, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka

ABSTRACT

This study investigates the impact of Large Language Models (LLMs) on problem-solving skills in source code debugging among IT undergraduates. A pre-, mid-, and post-experimental design was employed, including pre-test, mid-test, post-test (Prior), and post-test (Recent) phases to assess debugging performance with and without LLM assistance. The sample consisted of 87 students from the Department of Industrial Management, University of Kelaniya, Sri Lanka, stratified by gender, academic level, A/L stream, Z-score, and GPA. Results showed significant improvement in debugging accuracy, increasing from 46.53% in the pre-test to 69.51% in the post-test (Prior), indicating skill retention. Task efficiency also improved, with completion time reduced from 18 minutes to 10 minutes. However, transferability to new problems was moderate, with a post-test (Recent) accuracy of 58.40%. Higher academic levels, technical A/L streams, and mid-range GPAs were associated with better retention and adaptability. While LLMs enhanced immediate performance, the findings highlight the need to balance their use with independent practice to support long-term skill development. Limitations include resource constraints and short study duration, suggesting the need for longitudinal research. The study recommends structured integration of LLMs to optimize programming education outcomes.

ARTICLE HISTORY
Received 10 November 2025
Revised 30 March 2026
Accepted 7 April 2026

KEYWORDS
Large Language Models; debugging; programming education; problem solving skills; IT undergraduates

SUBJECTS
Arts & Humanities; Arts; Media Communication; Media Communication; Arts & Humanities; Humanities; Media & Film Studies; Media & Communications; Media Education; Social Sciences; Behavioral Sciences; Educational Psychology

Introduction

The increasing integration of Large Language Models (LLMs) in programming education has transformed how students approach coding and debugging tasks. Debugging is a critical skill for IT undergraduates, requiring logical reasoning, problem-solving, and technical proficiency (Whalley et al., 2021). While LLMs, such as ChatGPT, Google Gemini and GitHub Copilot, provide real-time code suggestions and debugging assistance, their impact on students' independent problem-solving skills remains an area of debate (Kasneci et al., 2023).

This study evaluates the influence of LLMs on the problem-solving skills of IT undergraduates in debugging software programming. As debugging is an essential competency in software development, it is crucial to determine whether reliance on LLMs enhances or diminishes students' ability to identify and resolve coding errors independently. Existing literature highlights both the benefits and challenges of using LLMs in higher education. While LLMs can improve efficiency and reduce cognitive load (Essel et al., 2024), concerns persist regarding potential over-reliance and reduced critical thinking development (Bubeck et al., 2023).

CONTACT PPG Dinesh Asanka dasanka@kln.ac.lk Department of Industrial Management, Faculty of Science, University of Kelaniya, Kelaniya, Sri Lanka.

To address these concerns, this research adopts a pre-test, mid-test, and post-test experimental approach to assessing the debugging performance with and without LLM assistance. The study involves undergraduate students from the University of INDUSTRIAL MANAGEMENT, stratified by academic level, GPA, and prior experience. By analyzing accuracy, efficiency, and adaptability in debugging tasks, this study aims to provide empirical insights into how LLMs influence skill retention and transferability in programming education.

The findings of this study contribute to understanding the role of AI in IT education and offer recommendations for the responsible integration of LLMs into programming curricula. This research will help educators maintain a balance between leveraging LLMs for enhanced learning and ensuring that students develop the fundamental problem-solving skills necessary for professional success. Overall, this work aims to answer the following research questions:

RQ1: How do LLMs influence debugging accuracy, efficiency, and the ability of IT undergraduates to identify errors in code?

RQ2: What differences exist in the debugging performance of IT undergraduates when using LLMs compared to debugging without LLM assistance, and how do these differences impact learning outcomes?

RQ3: Is there a statistically significant difference in debugging performance (with and without LLMs) across demographic categories such as gender, academic level, A/L stream, Z-score, and GPA?

Literature review

The use of Large Language Models (LLMs), such as OpenAI's GPT-3 and GPT-4, in educational settings, particularly in Computer Science and Information Technology, has garnered significant interest in recent years. These models have shown promise in aiding learners and professionals by providing real-time support for problem-solving tasks, including code debugging. This section reviews relevant literature on the role of LLMs in enhancing problem-solving skills, with a focus on their impact on programming tasks and code debugging. The review identifies the importance of a well-balanced integration of LLMs in education to maximize their benefits while addressing the challenges they present. Recommendations for future research directions are suggested to further explore the long-term impacts of LLMs on the development of problem-solving skills and their applicability across diverse academic environments (Riztha et al., 2024).

Importance of problem-solving skills in undergraduate education

Problem-solving skills are essential for both academic success and professional development, especially at the undergraduate level. These skills help students tackle complex tasks, make informed decisions, and apply theoretical knowledge to practical situations. In higher education, fostering independent problem-solving is crucial for preparing students for careers that require critical thinking, analysis, and evaluation of information to solve real-world problems. Research shows that undergraduates with strong problem-solving skills are better equipped for career success, as they demonstrate enhanced cognitive and practical abilities (Utami et al., 2019).

The development of problem-solving skills is also linked to other key attributes like assertiveness. Studies indicate that students who are more assertive tend to excel in problem-solving, boosting their confidence and academic performance (Güven, 2010). Universities are increasingly focusing on integrating problem-solving models into their curricula, as these structured approaches enhance critical thinking and help produce well-rounded graduates ready to meet the demands of the modern workforce (Utami et al., 2019).

The role of Large Language Models in education

The development of LLMs represents a significant milestone in Artificial Intelligence (AI) technology, offering unprecedented capabilities in generating human-like text, assisting with complex tasks, and enhancing learning experiences. In education, LLMs are being utilized as tools to facilitate learning by providing immediate answers, generating ideas, and even guiding students through problem-solving processes. These models are particularly valuable for helping students approach difficult subjects, but their role in education is still under research regarding their long-term impact on cognitive skill development (Kasneci et al., 2023).

LLMs are also contributing to fields such as language learning and writing by providing real-time feedback, generating writing prompts, and offering language translation services. Their role in enhancing critical thinking skills is evident across various disciplines, encouraging students to engage with complex concepts and analyze diverse viewpoints (Essel et al., 2024). Additionally, LLMs are being utilized in personalized learning environments, tailoring content to individual student needs and promoting more customized learning experiences. Their integration into educational practices underscores their transformative potential in shaping the future of education, positioning them as key tools in modern pedagogical strategies (Dong et al., 2024).

Evaluating LLMs' impact on programming education

While there is a growing body of research on the use of LLMs in programming education, studies specifically evaluating their impact on SRI LANKA undergraduates remain scarce. Research conducted by Wijesinghe et al. (2020) in the context of SRI LANKA's higher education system indicates that programming education faces several challenges, including inadequate resources and limited access to real-time assistance. This presents a unique opportunity for LLMs to play a transformative role by providing scalable and personalized learning support. As noted by Fernando (2020), the integration of LLMs in educational settings in SRI LANKA could help bridge gaps in programming skills development and provide a more efficient means for students to engage with programming tasks.

Significance of programming code debugging tasks in IT education and industry

Debugging is a critical aspect of IT education, essential for developing programming skills and fostering logical thinking, analytical reasoning, and problem-solving. It involves identifying and resolving code errors systematically, which enhances students' ability to detect issues, understand error patterns, and apply effective solutions. Whalley et al. (2021) highlight that debugging tasks encourage learners to analyze error symptoms and develop hypotheses, cultivating systematic thinking and improving code quality. However, teaching debugging presents challenges, as novice learners often struggle to adopt structured approaches due to limited exposure to error types and real-world scenarios.

In the professional IT domain, debugging is a key skill for software development and maintenance, with developers spending more time debugging than writing new code (Usmanova, 2023). Employers prioritize graduates proficient in debugging, as modern software environments become increasingly complex. Advanced debugging tools, such as the CodeT5-DLR framework, automate tasks like bug detection and repair (Bui et al., 2022). Additionally, Large Language Models (LLMs) show promise in bridging gaps in debugging skills, supporting both new and experienced programmers in real-world applications.

The integration of LLMs into the education of programming and code debugging presents an innovative approach to enhancing problem-solving skills. While the models have shown promise in aiding students' understanding of programming concepts and improving debugging efficiency, it is essential to evaluate their long-term impact on cognitive development and skill retention. This study seeks to address this gap by assessing the effectiveness of LLMs in problem-solving skills among SRI LANKAZ IT undergraduates, focusing on their ability to debug programming code effectively.

Transfer of learning

Transfer of learning plays a critical role in programming education, as students must apply previously acquired conceptual and procedural knowledge to novel coding and debugging tasks. Foundational theories distinguish between near and far transfer, emphasizing the importance of contextual similarity and abstraction in successful knowledge application (Barnett & Ceci, 2002). In programming contexts, effective debugging requires more than syntactic recall; it depends on the ability to transfer problem-solving strategies across varying code structures and error types (Robins et al., 2003). Recent research further highlights that transfer is influenced by instructional design, scaffolding, and learning environments, including transitions between programming paradigms and representations (Strong et al., 2025). Moreover, emerging studies suggest that AI-supported programming environments may influence how knowledge is internalized and transferred, particularly in collaborative or AI-assisted coding contexts (Welter et al., 2025). Together, these perspectives suggest that improvements observed across debugging tasks may reflect underlying transfer processes shaped by both pedagogical structure and technological support.

Methodology

This research employed a pretest–posttest experimental design, focusing on a group of undergraduate students from the University of Kelaniya, Sri Lanka specifically from the Department of Industrial Management IT's program. The methodology was designed to systematically evaluate the impact of Large Language Models (LLMs) on students' problem-solving skills in programming code debugging tasks. The primary objective was to assess how participants' debugging skills evolved through structured assessments conducted before, during, and after interacting with LLMs. The same approach was carried out by different research to Learning Effectiveness of Generative AI for Software Development (Samarakoon et al., 2024).

The four metrics employed in this study represent theoretically grounded dimensions of problem-solving skill development in programming. The cognitive dimension, operationalized through the number of errors identified, reflects learners' analytical reasoning and diagnostic abilities, as debugging requires understanding program logic, detecting discrepancies, and forming hypotheses about faults (Newell & Simon, 1972). The procedural dimension, measured by the number of errors correctly fixed, captures the application of conceptual and syntactic knowledge to implement effective solutions, reflecting procedural fluency and knowledge application in authentic contexts (Robins et al., 2003). The performance quality dimension, represented by the accuracy rate, provides an integrated measure of successful problem resolution by normalizing detection and correction performance, thereby indicating the quality and correctness of solutions. Finally, the efficiency dimension, measured through task duration, reflects cognitive fluency and expertise development, as increased proficiency is typically associated with reduced cognitive load and more efficient problem-solving processes (Ericsson et al., 1993; Sweller, 1988). Together, these dimensions provide a multidimensional assessment of debugging proficiency, capturing not only correctness but also depth of understanding and efficiency of execution.

The study followed a structured sequence, beginning with problem identification. The research aimed to address the challenge of enhancing problem-solving skills in programming code debugging tasks among IT undergraduates using LLM technology. After defining the research problem, an extensive literature review was conducted to examine existing studies on LLMs and their influence on debugging skills, particularly in educational contexts. This review established the theoretical foundation for the study and guided the research design.

Following the literature review, the research design was developed, incorporating both pre-assessment and post-assessment stages to evaluate participants' debugging performance. As shown in the Figure 1, The pre-assessment established baseline skills, while the post-assessment measured improvements following exposure to LLMs. Additionally, a mid-assessment phase was included to analyze how participants leveraged LLMs in real-time to enhance their problem-solving approaches.

The experimental process consisted of multiple assessment phases as shown in Table 1.

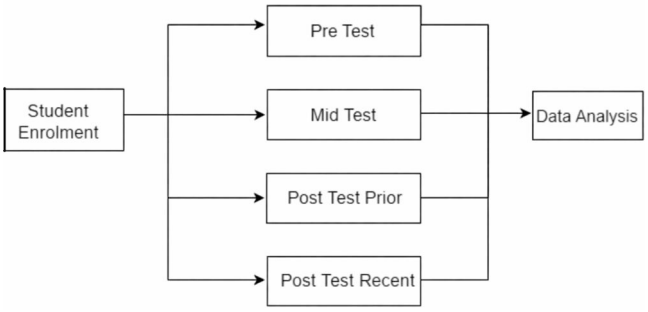

Figure 1. Pre-test, mid-test and post-test.

Table 1. Evaluation criteria.

Test type	Description	Purpose
Pre-Test	Students complete a given question without using Large Language Models (LLMs).	Establish baseline problem-solving skills without LLM assistance.
Mid-Test	Students answer the same question, this time with the assistance of LLMs.	Measure the impact of LLMs on students' ability to solve the same problem.
Post-Test (Prior)	Students attempt the same question again, but without using LLMs.	Assess knowledge retention and problem-solving ability without LLMs.
Post-Test (Recent)	Students answer a new but similar question without using LLMs.	Evaluate transfer of learning and adaptability to new but related problems.

The assessment tasks used in this study were structured debugging activities designed to evaluate students' analytical reasoning, procedural knowledge, and problem-solving efficiency within authentic programming contexts. The Pre-Test required students to identify and correct logical and syntax errors in a flawed Java-based E-Healthcare Management System, which involved object-oriented constructs such as classes, methods, collections, income calculations, and input handling. Students were expected to diagnose issues related to incorrect variable declarations, improper method definitions, logical miscalculations, and flawed control flow within a time constraint. The Post-Test (Recent) similarly required students to debug a Java-based University Grading System, where they needed to correct errors affecting GPA calculation, credit validation, classification logic, and data handling structures. Both assessments required learners to systematically identify faults, interpret program logic, and implement accurate corrections without external assistance, thereby providing comparable, scenario-based measures of debugging proficiency and higher-order problem-solving skill development.

Data were collected as part of a regular course assignment. Students were informed that participation in the study was voluntary and that only aggregated results would be used for research and publication purposes, with no individual-level data disclosed. Informed written consent was obtained accordingly. The pre-assessment phase established a baseline by evaluating participants' debugging skills without LLM assistance. The mid-assessment phase allowed students to interact with LLMs while debugging, providing real-time insights into their effectiveness in guiding problem-solving processes. The post-assessment phase was divided into two stages: in the post-assessment old phase, students debugged the same scenario as in the pre-assessment but without LLM assistance, testing their ability to retain and apply learned strategies. The post-assessment new phase introduced a novel debugging scenario, requiring students to solve a different programming issue independently. This final assessment evaluated whether the problem-solving strategies acquired during the LLM-assisted phase could be generalized to new contexts.

To ensure a diverse and representative sample, approximately 100 undergraduate students were selected based on stratification factors such as gender, academic level, A/L stream (The University entrance exam), Z-score of the A/L, overall GPA, preferred LLM, and previous programming module results that was conducted in the previous semester. This subject contains basic programming concepts such as lists, array, data types, selections and loops etc. This stratification accounted for variations in student backgrounds and enabled an analysis of how different factors influenced debugging performance and LLM-assisted learning outcomes.

The data collection process included both demographic data gathering and structured assessments. Demographic information was collected via an online questionnaire before the experimental phase. Debugging tasks were designed to align with participants' academic levels, with first-year students working in C++ and higher-level students working in Java. The debugging scenarios were based on real-world applications, with the initial three assessments focusing on a healthcare management system and the final assessment using a university grading system. Each assessment measured problem-solving skills, efficiency, and accuracy, with a 30-minute time limit per task.

All assessments were conducted under controlled conditions to ensure consistency and eliminate external influences. Participants were allowed to use their preferred IDEs, simulating real-world programming environments. Data from all four assessment phases (pre-test, mid-test, post-test (Prior) and post-test (Recent) were analyzed using statistical tools to determine the impact of LLMs on debugging performance, skill retention, and adaptability. This methodology provided empirical insights into the role of LLMs in programming education and contributed to understanding their benefits and limitations in fostering independent problem-solving skills.

Evaluation and scoring

The evaluation and scoring framework for this research was designed to objectively measure participants' debugging performance across four structured assessments. By capturing key quantitative metrics, this framework provided a comprehensive understanding of how Large Language Models (LLMs) influenced problem-solving skills in programming code debugging tasks. The evaluation process systematically tracked participants' performance, ensuring a robust analysis of their debugging capabilities, efficiency, and learning progress.

To assess the impact of LLMs, multiple performance indicators were recorded at each stage of the assessment. One of the primary metrics was the number of errors identified, which measured the participants' ability to thoroughly analyze the provided code and detect logical and syntactical issues. This metric highlighted the effectiveness of their debugging skills before and after LLM intervention. Additionally, the number of errors correctly fixed served as an indicator of problem resolution skills, reflecting participants' ability to apply appropriate debugging strategies. By evaluating this metric across different phases, the study examined whether exposure to LLMs enhanced the ability to resolve coding errors independently.

A distinction was made between logical and syntactical errors. The number of logical errors fixed was a critical measure of participants' ability to comprehend and correct flaws in the program's logic or algorithm, demonstrating their higher-order cognitive skills in programming. On the other hand, the number of syntax errors fixed evaluated participants' proficiency in identifying and correcting structural issues in the code that prevented proper execution. By analyzing these two types of errors separately, the study gained deeper insights into whether LLMs contributed more significantly to logical reasoning or fundamental syntax correction.

Efficiency was another key aspect of evaluation. The duration metric recorded the total time taken by participants to complete each debugging task. This measure provided valuable insights into whether LLM assistance contributed to increased efficiency and reduced debugging time. Comparing time efficiency across the pre-assessment, mid-assessment, and post-assessment phases allowed the study to assess the extent to which LLMs accelerated the debugging process while maintaining or improving accuracy.

Finally, the accuracy rate served as a comprehensive measure of overall debugging performance. It was calculated as the proportion of correctly fixed errors to the total number of errors in the code provided, expressed as a percentage. The accuracy rate provided a standardized way to compare participants' debugging effectiveness across different phases of the experiment.

$$Accuracy\ rate\ (\%) = \left(\frac{Number\ of\ Errors\ Correctly\ Fixed}{Total\ Errors\ in\ the\ Code}\right) \times 100$$

By integrating these key performance indicators, the evaluation framework ensured a systematic and objective assessment of participants' debugging skills. The analysis of these metrics not only highlighted the effectiveness of LLM-assisted debugging but also provided critical insights into whether LLMs contributed to long-term skill retention and adaptability in debugging tasks. To ensure the reliability and

validity of the evaluation framework, a mock test was conducted with a sample of five participants prior to the main assessments. This preliminary test helped refine the assessment process, ensuring that the designed metrics accurately captured participants' debugging performance.

Data analysis

Participants demographic overview

This study involved a total of 87 undergraduate participants from the Department of Industrial Management at the University of Kelaniya, spanning three academic levels: Level 1, Level 2, and Level 4. The participant demographics were analyzed across multiple dimensions, including gender, academic level, A/L stream, overall GPA, preferred LLM usage, and programming module results. The diversity in the sample ensured a comprehensive evaluation of how various factors influenced debugging performance and the impact of LLMs in programming education.

Gender Distribution: The participant sample was nearly balanced in terms of gender representation, with 53% male students (46 participants) and 47% female students (41 participants). This distribution allowed for a meaningful analysis of potential gender-based differences in debugging performance and the effectiveness of LLM-assisted learning.

Academic Level Distribution: The majority of participants belonged to Level 1 (52%), followed by Level 2 (32%) and Level 4 (16%). This distribution ensured the study captured a broad spectrum of debugging skills and experiences. The significant representation of Level 1 students provided insights into how LLMs influenced early-stage learners, while the inclusion of Level 2 and Level 4 students allowed for an in-depth analysis of skill development at more advanced academic stages.

A/L Stream Distribution: The participants' A/L backgrounds were predominantly in Physical Science (62%), followed by Bio Science (23%) and Physical Science with IT (15%). This distribution reflected a strong technical foundation among the majority of participants, which is directly relevant to programming and debugging tasks. The inclusion of students from diverse A/L streams enabled an analysis of how different educational backgrounds influenced problem-solving skills.

Overall GPA Distribution: The GPA distribution indicated that 48.28% of participants fell within the 3.0–3.5 range, representing a strong academic performance. Additionally, 19.54% had GPAs between 2.5 and 3.0, while 14.94% were in the 3.5–4.0 range, representing the highest academic achievers. This diverse range ensured a balanced representation of varying academic capabilities, allowing for an analysis of how GPA correlated with debugging performance and the effectiveness of LLM-assisted learning.

Preferred LLM Usage Distribution: The majority of participants (72.41%) preferred ChatGPT as their primary LLM tool for debugging tasks. Copilot was the second most used tool (12.64%), followed by Gemini (10.34%). Other LLMs, including Llama, Bolt AI, Blackbox.ai, and Perplexity, collectively accounted for 11.49% of participants. This variation in tool adoption provided an opportunity to examine differences in debugging performance and problem-solving strategies based on the preferred LLM.

Programming Module Result Distribution: The participants' academic performance in programming-related modules varied, with the largest proportion achieving a grade of A (21.84%). This was followed by B (17.24%), A- (14.94%), and B- (12.64%), collectively representing a significant portion of students with above-average performance. The presence of high-achieving students alongside those with moderate to lower grades ensured that the study could analyze how LLMs supported learners across different skill levels. Distributions of A/L Streams, GPA, LLM Usage, and Programming Results among Participants are shown in the Figure 2.

By examining these demographic factors, this study provided a holistic understanding of the participants' backgrounds, allowing for a deeper investigation into the effectiveness of LLMs in debugging tasks across different student profiles.

Results of key metrics for assessment phases

The study assessed participants' debugging skills across four different phases: Pre-Test, Mid-Test, Post-Test (Prior), and Post-Test (Recent). The results highlight significant improvements when using Large

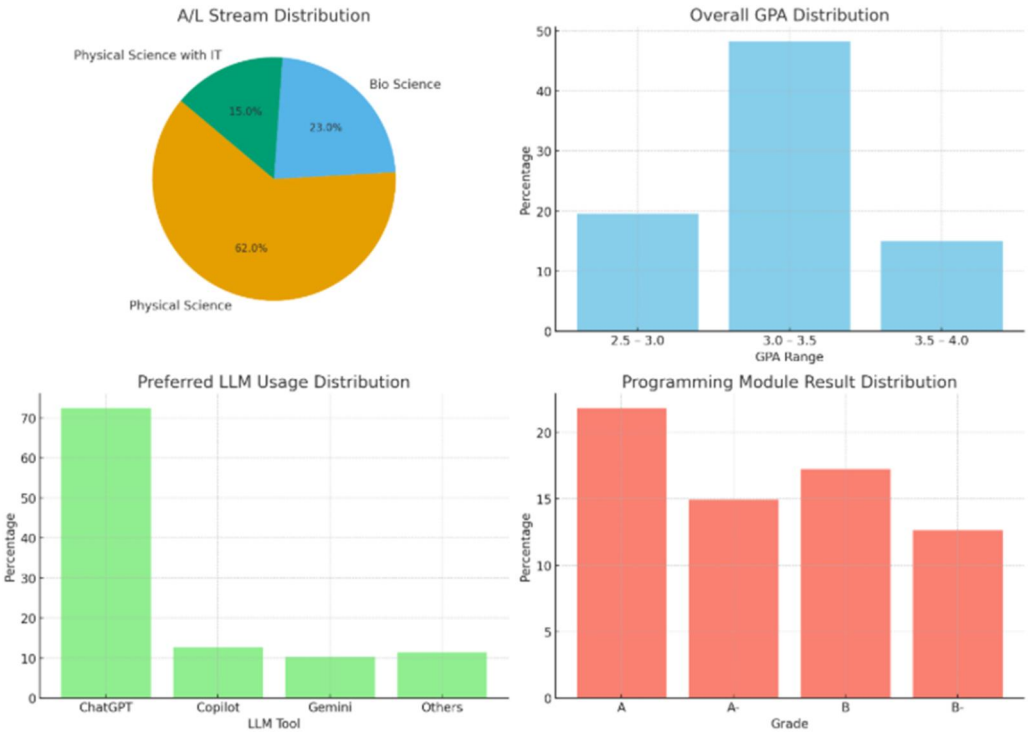

Figure 2. Distributions of A/L streams, GPA, LLM usage, and programming results among participants.

Language Models (LLMs) while also revealing challenges in retaining and generalizing debugging skills over time.

One of the key metrics measured was the participants' ability to locate errors in the given code. In the Pre-Test phase, participants identified an average of 43.75% of total errors, relying solely on their existing debugging skills. This figure dramatically increased to 100.00% in the Mid-Test phase when LLM assistance was introduced, demonstrating the powerful error-detection capabilities of AI-driven tools. However, after LLM exposure was removed, independent error identification dropped to 62.50% in the Post-Test (Prior) phase, indicating partial skill retention. When participants encountered a new and unfamiliar code scenario in the Post-Test (Recent) phase, the identification rate declined further to 56.25%, suggesting difficulties in generalizing acquired debugging skills to novel contexts. Refer, Figure 3.

Error correction was another crucial measure of debugging proficiency. In the Pre-Test phase, participants independently fixed 37.50% of errors. With LLM assistance in the Mid-Test phase, they achieved a 100.00% success rate, fully leveraging AI support. However, after LLM usage was withdrawn, the Post-Test (Prior) phase showed a reduction to 56.25%, indicating that although some debugging strategies were retained, participants struggled to replicate AI-assisted accuracy. The Post-Test (Recent) phase further declined to 50.00%, indicating a challenge in applying previously acquired knowledge to a fresh problem.

The study also examined logical and syntax error correction separately. Logical errors were particularly challenging, with only 16.67% fixed in the Pre-Test phase. The Mid-Test phase saw a dramatic jump to 100.00% due to LLM intervention. However, post-LLM exposure, the ability to fix logical errors, dropped to 50.00% in the Post-Test (Prior) phase and further declined to 33.33% in the Post-Test (Recent) phase, indicating significant difficulties in maintaining independent logical reasoning skills.

For syntax errors, participants fared relatively better. In the Pre-Test phase, they successfully fixed 50.00% of syntax errors, likely due to the more straightforward nature of syntax issues compared to logical errors. With LLM assistance, the Mid-Test phase again reached 100.00%. In the Post-Test (Prior) phase, participants retained a higher level of proficiency, fixing 70.00% of syntax errors, while in the Post-Test (Recent) phase, performance dropped to 60.00%, reflecting a moderate decline but still demonstrating a level of skill generalization.

Figure 3. Comparison of debugging performance metrics across tests.

Efficiency was also a key metric, measured by task duration Participants initially took an average of 18 minutes to complete debugging in the Pre-Test phase. The introduction of LLMs in the Mid-Test phase significantly reduced the completion time to just 8 minutes, demonstrating the time-saving benefits of AI assistance. Post-LLM exposure, participants in the Post-Test (Prior) phase further improved their efficiency, completing the task in 6 minutes. However, when faced with an unfamiliar code scenario in the Post-Test (Recent) phase, the task duration increased to 10 minutes, suggesting a reliance on familiarity when applying debugging strategies.

Finally, the accuracy rate, defined as the percentage of errors correctly fixed, further illustrates the impact of LLMs on debugging performance. In the Pre-Test phase, participants achieved an accuracy rate of 46.53%, reflecting their baseline debugging skills. The Mid-Test phase, with AI support, boosted accuracy to 100.00%. Post-LLM exposure, accuracy rates dropped to 69.51% in the Post-Test (Prior) phase, demonstrating partial skill retention. In the Post-Test (Recent) phase, accuracy declined further to 58.40%, reinforcing the challenge of transferring learned debugging techniques to unfamiliar scenarios.

A one-way analysis of variance (ANOVA) was conducted to examine differences in debugging accuracy across the three testing conditions: Pre-Test, Post-Test (Prior), and Post-Test (Recent). The results indicated a statistically significant difference in mean accuracy scores among the three groups, $F(2, 87) = 25.55$, $p < .001$. This finding suggests that problem solving skills varied significantly across the testing phases, indicating that the intervention conditions had a measurable impact on students' accuracy rates.

A Tukey HSD post-hoc test was conducted to determine which specific groups differed following the significant one-way ANOVA. The analysis revealed that all pairwise comparisons were statistically significant at the 0.05 level. Problem solving accuracy in the Post-Test (Prior) condition was significantly higher than in the Pre-Test condition (mean difference $= 0.233$, $p < .001$). Similarly, the Post-Test (Recent) condition demonstrated significantly higher accuracy compared to the Pre-Test condition (mean difference $= 0.111$, $p = .002$). Furthermore, a significant difference was observed between the two post-test conditions, with Post-Test (Prior) yielding higher accuracy than Post-Test (Recent) (mean difference $= 0.122$, $p = .001$).

Overall, the findings indicate that while LLMs significantly enhance debugging performance by improving error detection, correction accuracy, and efficiency, they also present challenges regarding skill retention and generalization. Although participants retained some debugging proficiency after LLM exposure, their performance declined when encountering new problems, suggesting a reliance on AI-generated solutions. This underscores the importance of balancing AI-assisted learning with independent problem-solving practice to develop robust and transferable debugging skills.

Results with demographic variables

The analysis of demographic variables in the study of debugging performance with and without Large Language Models (LLMs) reveals several important trends based on gender, academic level, A/L stream, Z-scores, and GPA. In terms of gender, as shown in the Figure 4, the data showed minimal differences between male and female participants across all assessment phases. Female participants had a slightly higher baseline performance in the pre-test, with an accuracy rate of 48.28% compared to males at 44.89%. However, both genders achieved perfect accuracy in the mid-test when assisted by LLMs, indicating that LLMs effectively equalized performance across genders. The post-test (Prior) phase, measuring skill retention, showed that both males and females retained similar levels of skill, with males achieving 69.89% and females 69.11%. While there was a slight difference in the post-test (Recent) phase, where males demonstrated a marginal advantage in skill transferability (60.51% compared to 56.14% for females), the overall gender-based differences were minimal. This suggests that LLMs served as a universally beneficial tool for improving problem-solving abilities, with individual factors such as problem-solving approaches and adaptability playing a more significant role than gender.

The analysis based on academic level revealed (Figure 5) that participants from higher academic levels performed better in terms of baseline debugging accuracy. Level 1 participants had a baseline accuracy rate of 47.67%, while Level 4 participants outperformed all others with an accuracy rate of 52.54%. However, LLMs helped all participants achieve perfect accuracy in the mid-test phase, regardless of their academic level, underscoring the tool's universal effectiveness in enhancing debugging skills. In the post-test (Prior) phase, Level 4 participants exhibited the highest skill retention, achieving 82.49%, while Level 1 participants retained 69.41%. This difference suggests that students at higher academic levels were better able to internalize and apply the skills learned during the LLM-assisted phase. Similarly, in the post-test (Recent) phase, which tested the transferability of learned skills to novel debugging scenarios, Level 4 participants again demonstrated superior performance, with an accuracy rate of 64.42%. The findings highlight that academic experience and programming maturity contribute significantly to the adaptability and transferability of skills.

ANOVA was conducted to examine whether debugging accuracy in the Post-Test (Recent) condition differed across academic levels. The results revealed a statistically significant difference in accuracy among the different levels, $F(2, 87) = 3.32$, $p = .041$. These results indicate that students' Problem Solving performance in the recent post-test condition varied significantly depending on their academic level.

A Tukey HSD post-hoc analysis was conducted to identify which academic levels differed in debugging accuracy in the Post-Test (Recent) condition. The results indicated a statistically significant

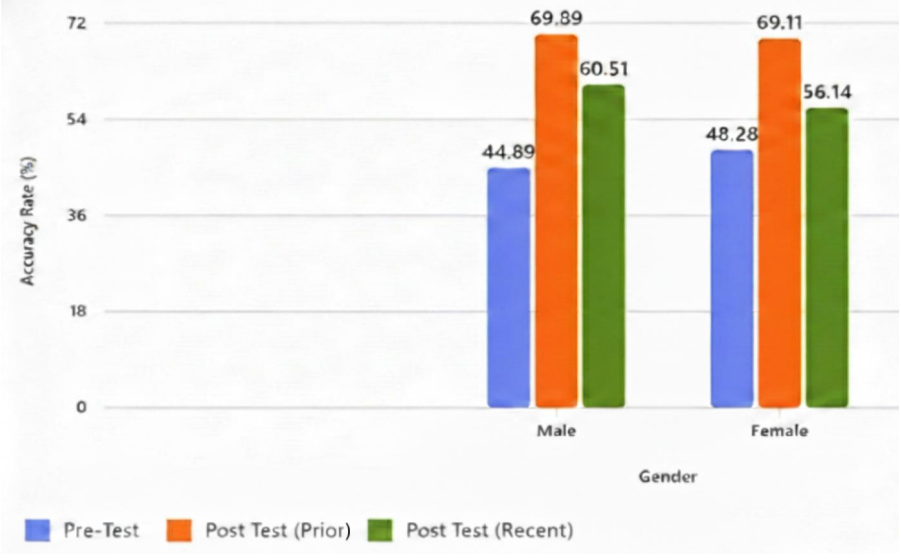

Figure 4. Comparison of accuracy rates across genders.

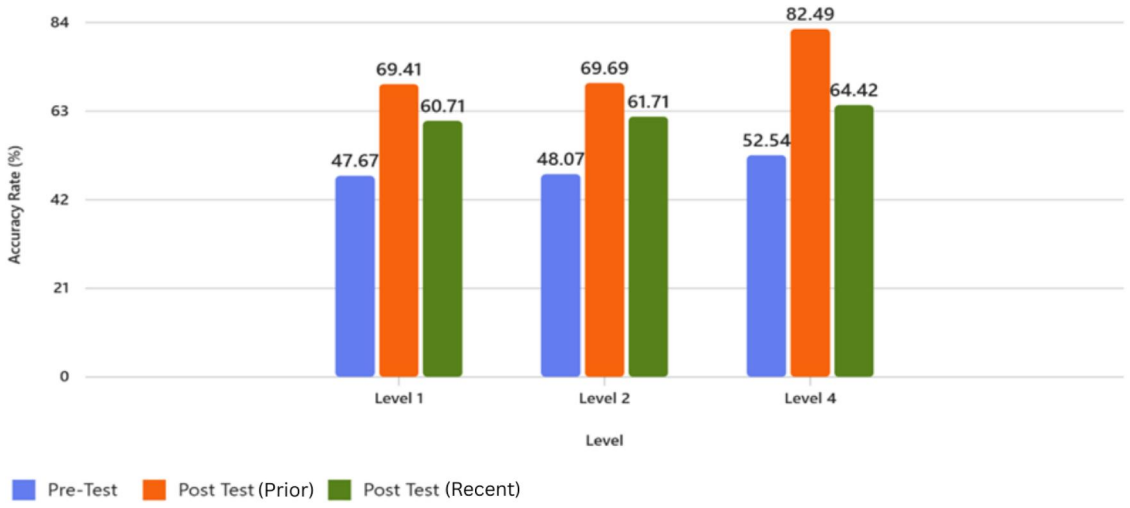

Figure 5. Comparison of accuracy rates across levels.

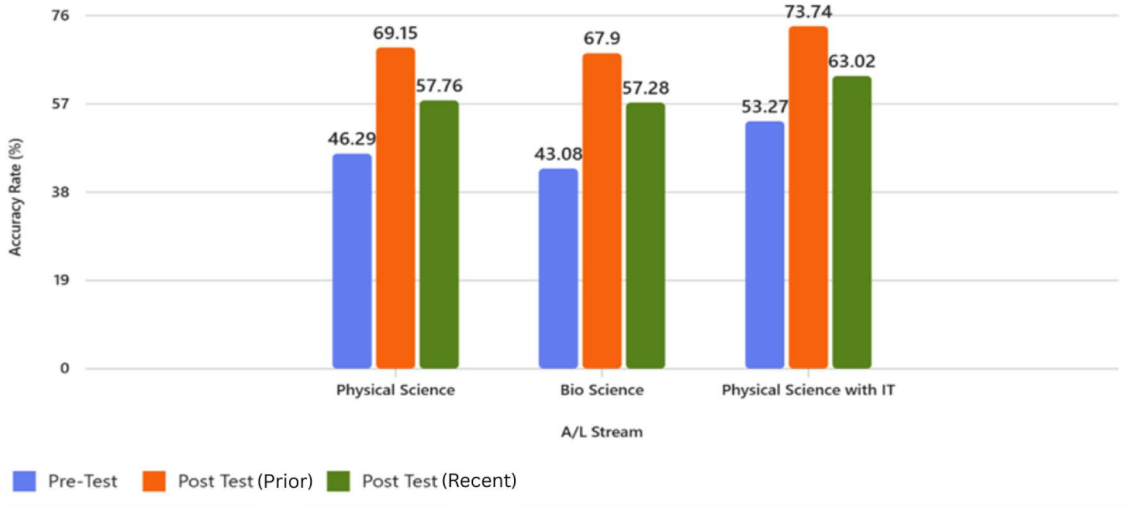

Figure 6. Comparison of accuracy rates across different A/L streams.

difference between Level 2 and Level 4 students (mean difference = 0.166, $p = .036$), with Level 4 students demonstrating higher accuracy.

These findings suggest that the significant ANOVA result was primarily driven by the difference between Level 2 and Level 4 students, while other pairwise comparisons did not show meaningful differences.

When analyzing participants by A/L stream, differences in baseline performance were observed, particularly between the streams of Physical Science, Bio Science, and Physical Science with IT, as shown in Figure 6. Participants from the Physical Science with IT stream had the highest baseline accuracy rate (53.27%), while those from the Bio Science stream had the lowest (43.08%). In the mid-test phase, all streams achieved perfect accuracy, demonstrating that LLMs effectively levelled the playing field for participants regardless of their academic background. However, in terms of skill retention and transferability, participants from the Physical Science with IT stream again outperformed the others. They achieved the highest retention rate in the post-test (Prior) phase (73.74%) and the best transferability in the post-test (Recent) phase (63.02%). This suggests that students with a more technical background were better equipped to retain and apply LLM-assisted strategies. These results underscore the importance of aligning educational interventions with students' technical backgrounds to maximize the benefits of LLMs.

The Z-score analysis shown in Figure 7, which categorized participants by performance improvement, revealed that those with mid-range Z-scores (0.5–1.0) showed the most significant improvement in

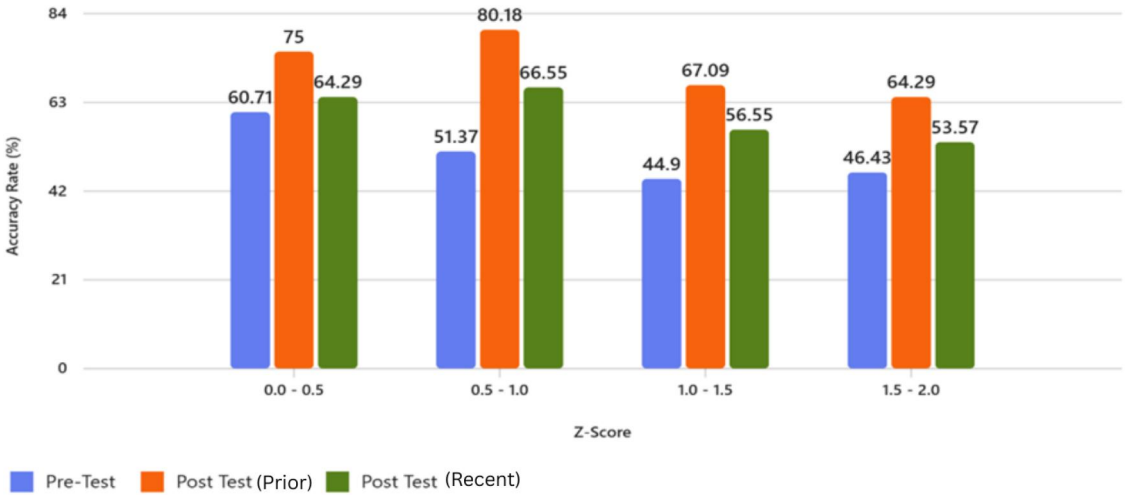

Figure 7. Comparison of accuracy rates across Z-scores.

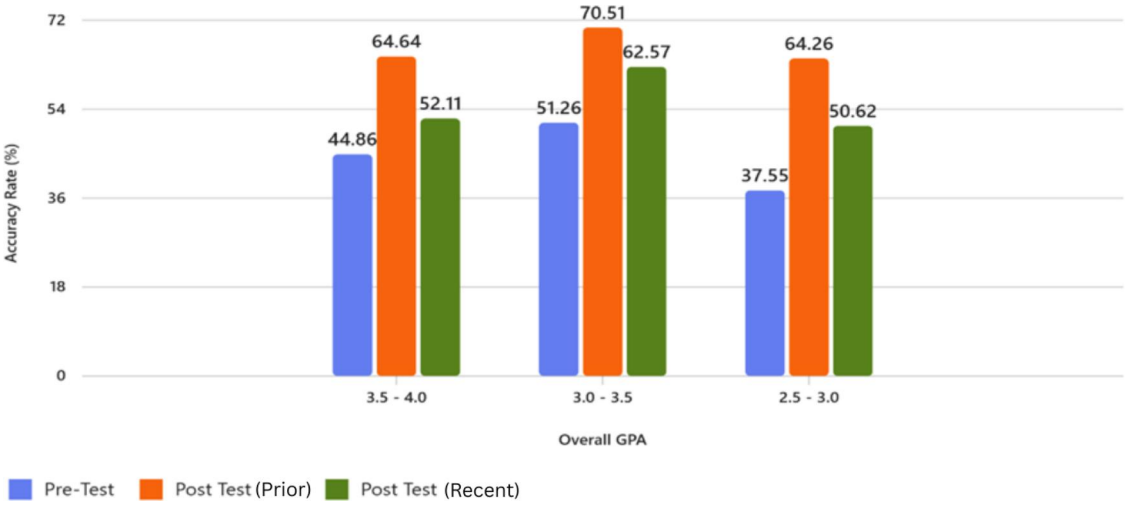

Figure 8. Comparison of accuracy rates across GPA.

debugging accuracy and skill transferability. These participants had the most significant increase in performance from pre-test to post-test (Prior), with a 28.81% improvement, and demonstrated the highest skill transferability to novel scenarios, with a 15.18% increase in accuracy in the post-test (Recent). On the other hand, participants with lower and higher Z-scores showed moderate improvements, suggesting that those with a balanced mix of foundational skills and cognitive flexibility benefited the most from LLM assistance. This finding highlights the potential of LLMs to enhance debugging skills, particularly for participants in the middle range of performance, where improvements in problem-solving abilities were most pronounced.

Lastly, the GPA-based analysis as shown in Figure 8, revealed that students with mid-range GPAs (3.0–3.5) exhibited the strongest debugging performance and skill transferability. These participants had the highest baseline accuracy (51.26%) in the pre-test phase and showed the most significant retention and adaptability in the post-test (Prior) and post-test (Recent) phases. In contrast, participants with higher GPAs (3.5–4.0) and lower GPAs (2.5–3.0) demonstrated lower retention and transferability. The findings suggest that mid-range GPA students may possess a balance of foundational skills and cognitive flexibility, enabling them to benefit more from LLM assistance. Overall, the GPA-based analysis emphasizes the importance of a balanced skill set for effective problem-solving, with students in the 3.0–3.5 GPA range showing the most notable improvements across all assessment phases.

In summary, the results of this demographic analysis provide valuable insights into how various factors such as gender, academic level, A/L stream, Z-scores, and GPA influence the effectiveness of LLMs in enhancing debugging skills. While LLMs proved beneficial across all demographic groups, factors such

Table 2. Accuracy rate of students with different demographics.

	Accuracy rate			
	Pre-test	Mid-test	Post-test (prior)	Post-test (recent)
Gender				
Male	44.89%	100%	69.89%	60.51%
Female	48.28%	100%	69.11%	56.14%
Level				
Level 1	47.67%	100%	69.41%	60.71%
Level 2	48.07%	100%	69.69%	61.71%
Level 4	52.54%	100%	82.49%	64.42%
A/L Stream				
Physical Science	46.29%	100%	69.15%	57.76%
Bio Science	43.08%	100%	67.90%	57.28%
Physical Science with IT	53.27%	100%	73.74%	63.02%
Z-Score				
0.0–0.5	60.71%	100%	75.00%	64.29%
0.5–1.0	51.37%	100%	80.18%	66.55%
1.0–1.5	44.90%	100%	67.09%	56.55%
1.5–2.0	46.43%	100%	64.29%	53.57%
Overall GPA				
3.5–4.0	44.86%	100%	64.64%	52.11%
3.0–3.5	51.26%	100%	70.51%	62.57%
2.5–3.0	37.55%	100%	64.26%	50.62%

as academic experience, technical background, and cognitive flexibility played crucial roles in determining the extent to which participants retained and transferred their learned skills. These findings suggest that LLMs have the potential to bridge performance gaps, but additional support may be required to maximize their impact on students from less technical backgrounds. Table 2 indicates the accuracy rate of students for different demographics such as Gender, Level, A/L stream, Z-score, and GPA.

Discussion

The results of this study suggest that the use of Large Language Models (LLMs) significantly enhances debugging performance, with participants showing notable improvement between the pre-test and post-test (Prior) phases. This indicates that LLMs can serve as effective educational tools for enhancing problem-solving skills in programming, even when not actively in use.

Improvement in Debugging Accuracy: The comparison between the pre-test and post-test (Prior) results highlights a substantial improvement in participants' ability to identify and resolve programming errors independently. The accuracy rate increased from 46.53% in the pre-test to 69.51% in the post-test (Prior), reflecting not only immediate gains but also the retention of skills over time. Participants exhibited enhanced abilities in detecting both logical errors (up from 16.67% in the pre-test to 50.00% in the post-test) and syntax errors (up from 50.00% in the pre-test to 70.00% in the post-test). These improvements suggest that LLMs assisted in internalizing debugging strategies and deepening participants' cognitive understanding of programming concepts.

Transferability of Skills to Novel Scenarios: In the post-test (Recent) phase, participants were presented with a novel debugging scenario to assess their ability to apply the skills learned to unfamiliar problems. The results revealed a moderate drop-in accuracy rate, from 69.51% in the post-test (Prior) to 58.40% in the post-test (Recent). Despite this decrease, the accuracy rate in the post-test (Recent) was still higher than the pre-test rate (46.53%), indicating that participants retained and applied some of the debugging skills learned through LLM interaction. While there is a noticeable gap in transferring skills to novel contexts, the partial transferability observed suggests that LLM-assisted learning provides a solid foundation for problem-solving.

Enhanced Debugging Efficiency: Participants demonstrated a significant reduction in the time taken to complete debugging tasks, from an average of 18 minutes in the pre-test to just 10 minutes in the post-test (Recent). This improvement in task efficiency suggests that participants became more adept at identifying and resolving errors quickly, even when faced with new and unfamiliar challenges. The reduction in debugging time is indicative of improved problem-solving speed and enhanced cognitive processing abilities developed through LLM-assisted practice.

Variations Across Demographics: The analysis of demographic variables revealed consistent improvements in debugging accuracy across various groups, including gender, academic level, A/L stream, district, and GPA. Notably, participants from higher academic levels (e.g. Level 4) and those from technical A/L streams (e.g. Physical Science with IT) exhibited the greatest improvements, with post-test (Prior) rates exceeding 70%. Conversely, participants from lower academic levels (e.g. Level 1) or non-technical streams (e.g. Bio Science) also showed meaningful progress, although their post-test results were somewhat lower compared to their more experienced peers. These findings suggest that while LLMs benefit all participants, those with a stronger foundational understanding of programming tend to show more pronounced improvements in debugging accuracy.

Overall, the results of this study demonstrate that LLMs can effectively enhance both immediate and long-term problem-solving abilities in programming, as evidenced by the significant improvements in debugging accuracy and efficiency. However, the partial transferability of skills to new contexts highlights a potential area for further development, suggesting the need for strategies that encourage independent learning and adaptability. Incorporating LLMs into educational practices can thus play a crucial role in advancing students' problem-solving capabilities, provided that learners are also guided to cultivate the skills necessary to apply their knowledge to new, unfamiliar challenges.

Challenges and limitations

This study encountered several challenges and limitations that could impact the generalizability and validity of the findings. The research was conducted within a short time frame, limiting the ability to assess the long-term effects of LLMs on skill retention and transferability. Additionally, resource constraints, including limited access to a range of LLM tools and a small sample size of 87 participants from a single university, may affect the broader applicability of the results. The study also faced issues with demographic representation, as participants from certain regions or educational backgrounds showed varied performance, highlighting the need for a more diverse participant pool in future research.

Other limitations included reliance on self-reported data, which may introduce bias, and a focus on specific programming scenarios, limiting the diversity of tasks tested. The study also did not evaluate the long-term impact of LLMs on problem-solving skills, and there was a risk of participants over-relying on LLMs during the mid-test phase, which may not reflect true skill development. These challenges suggest that further studies, with larger and more diverse samples, longitudinal assessments, and a broader range of programming tasks, would be necessary to enhance the validity and applicability of the findings.

Implications for practice

The findings of this research provide valuable insights into the integration of Large Language Models (LLMs) into programming education, highlighting how these tools can enhance debugging performance and skill retention. These implications can guide educators, institutions, and LLM developers in effectively implementing LLMs to maximize their educational potential.

1. **Enhancing Educational Strategies**: LLMs can be used to support programming education by providing real-time assistance and feedback. Educators can incorporate LLMs through structured tasks that help students identify and fix errors, gradually reducing LLM dependency to ensure independent skill development. Customizable learning pathways can also be employed, allowing LLMs to provide personalized support, catering to students' skill levels.
2. **Balancing LLM Use with Independent Practice**: While LLMs boost short-term performance, it's essential to balance their use with independent practice. Educators should emphasize critical thinking, problem-solving strategies, and introduce LLM-free assessments to foster internalization of learned skills. Training should also focus on guiding students to interpret LLM suggestions rather than merely copying them.
3. **Addressing Skill Transferability Challenges**: To help students apply their learned skills to novel problems, educators should expose them to diverse programming tasks and real-world scenarios.

Scenario-based learning, including industry-relevant projects, can also enhance adaptability and improve performance in unfamiliar contexts.

4. **Reducing Educational Inequities**: The study revealed disparities in performance based on geographic and demographic factors. Institutions should ensure equal access to LLM tools and programming resources, particularly in underrepresented areas, and provide workshops to familiarize students from non-technical backgrounds with LLM usage and debugging strategies.
5. **Institutional Policy Recommendations**: Institutions should develop policies that guide the responsible use of LLMs in programming curricula. Ethical guidelines should be established to prevent misuse, and assessments should combine LLM-assisted and independent tasks to evaluate both immediate and long-term skill development. Additionally, continuous professional development for educators will help them effectively integrate LLMs into teaching.
6. **Implications for LLM Developers**: Developers can use these findings to enhance LLM tools for education. Features that explain suggestions, provide alternative solutions, and guide students step-by-step could be beneficial. Additionally, introducing modes that reduce hints over time and offering feedback on specific types of errors can support skill development.

Finally, LLMs hold transformative potential in programming education, but their integration must be thoughtful to balance short-term gains with long-term skill development. By aligning LLM use with effective pedagogical strategies, addressing access disparities, and ensuring skill transferability, educators and institutions can create an equitable and effective learning environment that prepares students for real-world programming challenges.

Conclusion

This study provides valuable insights into the role of Large Language Models (LLMs) in programming education, particularly in enhancing debugging performance and problem-solving skills among Sri Lankan IT undergraduates. The findings demonstrate that LLMs improve debugging accuracy, efficiency, and skill retention, though challenges remain in applying learned skills to novel scenarios. Demographic and academic factors were found to influence the effectiveness of LLMs, with more advanced academic levels and technical backgrounds showing better results. The research contributes significantly to the field of educational technology by offering empirical evidence on LLMs' impact on debugging, examining demographic moderators, developing a framework for skill retention and transferability, and providing practical implications for integrating LLMs into curricula. It also emphasizes the importance of balancing LLM use with independent practice and addressing disparities in educational access, paving the way for future research and integration of LLMs into programming education.

Recommendations for future research

Future research should focus on addressing the limitations identified in this study and expanding its scope to provide a deeper understanding of the long-term impact of Large Language Models (LLMs) in programming education. Longitudinal studies are essential to evaluate how skills learned with LLM assistance evolve over time, assessing both retention and adaptability to real-world scenarios. Additionally, research should explore the role of LLMs in collaborative learning environments, where group-based problem-solving and team dynamics could further enhance debugging skills. Expanding the participant pool to include students from diverse academic disciplines, cultural backgrounds, and geographic regions will improve the generalizability of findings and shed light on how context influences LLM effectiveness. Comparative studies analyzing various LLM tools, such as ChatGPT, GitHub Copilot, and Gemini, can identify specific features that optimize learning outcomes. Lastly, future research should investigate strategies to mitigate over-reliance on LLMs, ensuring students develop independent critical thinking and problem-solving skills. These avenues will contribute to a comprehensive understanding of LLMs' potential and challenges, paving the way for their effective integration into programming education.

Concluding remarks

This study underscores the transformative potential of LLMs in programming education. By bridging gaps in debugging skills and enhancing problem-solving efficiency, LLMs can significantly improve learning outcomes when used effectively. However, the findings also emphasize the importance of balancing LLM use with independent practice to ensure students develop sustainable, transferable skills. As technology continues to reshape education, integrating tools like LLMs into curricula offers exciting opportunities to revolutionize teaching and learning. By addressing challenges such as over-reliance, resource disparities, and skill adaptability, educators and institutions can harness the full potential of LLMs to empower students and prepare them for the complexities of modern programming and problem-solving tasks.

Acknowledgements

The authors acknowledge the use of AI-based tool, ChatGPT for assistance in editing, grammar enhancement, and spelling check during the preparation of this manuscript. Conceptualization: Ruwan Wickramarachchi/PPG Dinesh Asanka; Methodology: Fathima Riztha/Ruwan Wickramarachchi/PPG Dinesh Asanka/Mathishi Adya Dissanayake; Data Curation & Validation: Fathima Riztha/Mathishi Adya Dissanayake; Project administration: Ruwan Wickramarachchi/PPG Dinesh Asanka; Original draft preparation: Fathima Riztha; Writing review and editing: PPG Dinesh Asanka/Mathishi Adya Dissanayake

Ethical approval

Formal ethical approval was not required for this study, as the data were collected as part of regular course activities within the curriculum of the Department of Industrial Management, University of Kelaniya, Sri Lanka. Students were informed about the study objectives and provided the option to participate voluntarily. Only aggregated, anonymized data were used for analysis, ensuring that no individual-level information was disclosed. Considering these facts, research ethic committee at the Department of Industrial Management, University of Kelaniya, Sri Lanka exempted the ethical approval. Details of the Ethical review https://im.kln.ac.lk/research

Author contributions

CRediT: **Fathima Riztha**: Data curation, Validation, Visualization, Writing – original draft; **Ruwan Wickramarachchi**: Methodology, Project administration, Supervision; **PPG Dinesh Asanka**: Conceptualization, Methodology, Project administration, Supervision, Validation, Writing – review & editing; **Mathishi Adya Dissanayake**: Investigation, Methodology, Software, Writing – review & editing.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

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

ORCID

Fathima Riztha http://orcid.org/0009-0005-6037-5781
Ruwan Wickramarachchi http://orcid.org/0000-0001-7004-8702
PPG Dinesh Asanka http://orcid.org/0000-0002-3433-5533
Mathishi Adya Dissanayake http://orcid.org/0009-0008-2993-5527

Data availability

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

References

Barnett, S. M., & Ceci, S. J. (2002). When and where do we apply what we learn? A taxonomy for far transfer. *Psychological Bulletin*, *128*(4), 612–637. https://doi.org/10.1037/0033-2909.128.4.612

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., & Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv. https://doi.org/10.48550/arXiv.2303.12712

Bui, N. D. Q., Wang, Y., & Hoi, S. C. H. (2022). Detect-Localize-Repair: A unified framework for learning to debug with CodeT5.In *Findings of the association for computational linguistics: EMNLP*, 2022 (pp. 812–823). https://doi.org/10.18653/v1/2022.findings-emnlp.57

Dong, B., Bai, J., Xu, T., & Zhou, Y. (2024, April). Large language models in education: A systematic review. In *Proceedings of the 2024 international conference on computer science and technology in education (CSTE)*. IEEE. https://doi.org/10.1109/cste62025.2024.00031

Essel, H. B., Vlachopoulos, D., Essuman, A. B., & Amankwa, J. O. (2024). ChatGPT effects on cognitive skills of undergraduate students: Receiving instant responses from AI-based conversational large language models (LLMs). *Computers and Education: Artificial Intelligence*, *6*, 100198. https://doi.org/10.1016/j.caeai.2023.100198

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, *100*(3), 363–406. https://doi.org/10.1037/0033-295X.100.3.363

Fernando, W. P. K. (2020). Information and communication technology in Sri Lankan higher education. *Journal of Research Technology and Engineering*, *1*(2), 1–3.

Güven, M. (2010). An analysis of the vocational education undergraduate students' levels of assertiveness and problem-solving skills. *Procedia - Social and Behavioral Sciences*, *2*(2), 2064–2070. https://doi.org/10.1016/j.sbspro.2010.03.282

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., … Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, *103*, 102274. https://doi.org/10.1016/j.lindif.2023.102274

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Riztha, F., Wickramarachchi, R., Asanka, D., & Disssanayke, M. (2024 Assessing the impact of large language models on problem-solving skills of undergraduates: A systematic literature review. In *2024 6th international conference on advancements in computing (ICAC)*, Colombo, Sri Lanka (pp. 408–413). IEEE. https://doi.org/10.1109/ICAC64487.2024.10850942

Samarakoon, P., Asanka, D., Jayalal, S., & Jayalath, N. (2024). *Analyzing the learning effectiveness of generative AI for software development for undergraduates in Sri Lanka*.In *2024 international research conference on smart computing and systems engineering (SCSE)*, Colombo, Sri Lanka (pp. 1–7). IEEE. https://doi.org/10.1109/SCSE61872.2024.10550837

Strong, G., Bresnihan, N., & Tangney, B. (2025). Supporting learners in the transition from block-based to text-based programming: A systematic review. *Journal of Computer Languages*, *84*, 14–26.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Utami, B., Probosari, R. M., Saputro, S., Ashadi, N., & Masykuri, M. (2019). Empowering critical thinking skills with problem solving in higher education. *Journal of Physics: Conference Series*, *1280*(3), 032047. https://doi.org/10.1088/1742-6596/1280/3/032047

Welter, A., Schneider, N., Dick, T., Weis, K., Tinnes, C., & Wyrich, M. (2025). From developer pairs to AI copilots: A comparative study on knowledge transfer. arXiv. https://doi.org/10.48550/arXiv.2506.04785

Whalley, J., Settle, A., & Luxton-Reilly, A. (2021). *Analysis of a process for introductory debugging*.In *Proceedings of the 23rd Australasian computing education conference (ACE '21)*, 2–4 February 2021, Virtual, SA, Australia (pp. 45–54). ACM. https://doi.org/10.1145/3441636.3442300

Wijesinghe, C., Hansson, H., Ekenberg, L., & Hettiarachchi, E. (2020). Influential factors for ICT Innovations in Sri Lanka University-Industry Collaboration: A systematic literature review. *Journal of Research Innovation and Implications in Education*, *4*(4), 47–65.

Usmanova, D. (2023). Developer productivity: Reducing time spent on debugging Undo*io*. https://undo.io/solutions/developer-productivity/reduce-time-spent-debugging/