International Journal of Computing Sciences Research (ISSN print: 2546-0552; ISSN online: 2546-115X)
Vol. 10, pp. 4266-4291
doi: 10.25147/ijcsr.v10i0.837
https://stepacademic.net

Long Paper

A Framework for Enhancing Personalized Learning of Computer Programming using Large Language Models

Erick Odhiambo Omuya
Department of Computing and Information Technology, Machakos University, Kenya
omuya.erick@mksu.ac.ke
ORCID: 0000-0002-1646-397X
(corresponding author)

Fredrick Muema Mboya
Department of Information Technology, Murang'a University of Technology, Kenya
muemafred72@gmail.com

Geoffrey Mariga Wambugu
Department of Information Technology, Murang'a University of Technology, Kenya
gmariga@mut.ac.ke
ORCID: 0000-0001-6899-1152

Joyce Wangui Gikandi
Department of Educational Management and Curriculum Studies, Mount Kenya University, Kenya
wagitash@gmail.com
ORCID: 0000-0001-8531-6212

Faith Mueni Musyoka
Department of Computing and Information Technology, University of Embu, Kenya
mueni.faith@embuni.ac.ke
ORCID: 0000-0002-9574-8235

Date received: February 20, 2026
Date received in revised form: May 22, 2026; June 19, 2026
Date accepted: June 26, 2026

Recommended citation:

Omuya, E. O., Mboya, F. M., Wambugu, G. M., Gikandi, J. W., & Musyoka, F. M. (2026). A framework for enhancing personalized learning of computer programming using large language models. *International Journal of Computing Sciences Research*, *10*, 4266-4291. https://doi.org/10.25147/ijcsr.v10i0.837

Abstract

Purpose – This study presents a framework that utilizes prompt engineering to refine AI-generated content, ensuring its coherence, accuracy, and educational relevance. It explores the integration of LLMs to enhance personalized learning in introductory computer programming courses, focusing on the C programming language.

Method – The quasi-experimental design was adopted to compare traditional teaching methods with LLM-supported learning. We evaluated adaptive learning algorithms to track changes in student performance over time using metrics like reduced error rates, average time spent on tasks, and task completion rates.

Results – The experimental group recorded a pretest mean of 57.16 and an improved posttest mean of 66.24, yielding a gain of +9.08 marks. The control group began with a pretest mean of 60.94 and achieved a posttest mean of 63.08, resulting in a comparatively lower gain of +2.14 marks. To determine statistical significance between the groups, a Mann-Whitney U test was conducted on the posttest scores. The test yielded a U-value of 1,725.5, a Z-score of -2.634, and a p-value of 0.008, indicating that the difference between the experimental and control groups was statistically significant at the $p < 0.01$ level.

Conclusion – Effective prompt engineering strategies reduce hallucinations and align generated content with curriculum objectives, thereby enabling scalable, adaptive, and student-centered support in programming education.

Recommendations – Learning through LLM frameworks should adopt effective prompt engineering strategies to achieve a practical impact on student engagement, content understanding, and the overall learning experience.

Practical Research Implications – The findings reinforce the pivotal role of prompt engineering in LLM-based learning. Effective GenAI frameworks for education must go beyond novelty and automation and instead embed educationally grounded principles that support learners' development through intelligent, responsive, and context-aware interactions.

Keywords – large language models, prompt engineering, personalized learning, computer programming education, generative AI

INTRODUCTION

The rapid advancement of artificial intelligence (AI) has significantly impacted various educational domains, including computer programming (Zhai et al., 2021). The programming field is continuously evolving, with educators seeking innovative methods to enhance student engagement and learning outcomes (Durak, 2020). Despite the centrality of programming in computing education, students often report difficulties in maintaining engagement and achieving mastery, particularly in introductory courses. Factors contributing to low engagement include abstract concepts, a steep learning curve, and a lack of timely feedback or individualized support (Cheah, 2020; Chen et al., 2024). Traditional

teaching approaches, such as standardized assignments and lectures, often fail to accommodate diverse learning paces and styles, limiting students' ability to apply theoretical knowledge effectively (Wood and Moss, 2024). Personalized learning approaches have been used to address these challenges by tailoring content and support to individual needs. However, implementing such solutions at scale remains a significant challenge in programming education.

In recent years, the advent of large language models (LLMs) such as GPT-4 has opened new avenues for personalized learning experiences. These models, with their ability to generate human-like text, hold promise for providing tailored code examples and explanations to students, potentially transforming the traditional learning environment (Chen et al., 2024; Li et al., 2024). However, leveraging LLMs in educational contexts is not without challenges. One of the primary issues is the quality of the prompts given to these models. The effectiveness of LLMs largely depends on how well the prompts are constructed (Banjade et al., 2024; Cain, 2024; Maity et al., 2024). A well-designed prompt can lead to accurate, relevant, and coherent outputs, while a poorly designed prompt can result in irrelevant, misleading, or nonsensical responses (Patil et al., 2024; Szymanski et al., 2024). This highlights the critical role of prompt engineering in optimizing the interaction between students and LLMs.

Prompt engineering is the process of designing and refining prompts to elicit the desired output from LLMs (Marvin et al., 2024). It involves understanding the nuances of how these models interpret and respond to different types of inputs. Effective prompt engineering can significantly enhance the educational value of LLMs by ensuring that the generated content aligns with learning objectives and aids in comprehension (Cain, 2024; Wang et al., 2024). Good prompts should exhibit clarity, specificity, contextual relevance, and instructiveness. Clarity ensures that the model understands what is being asked, while specificity helps narrow down the possible responses. Contextual relevance ensures that the generated content is appropriate for the educational setting, and instructiveness provides clear guidance to the model (Sivarajkumar et al., 2024).

To scientifically measure the efficiency of prompts, various metrics can be employed. Accuracy and relevance can be evaluated using precision and recall metrics, while coherence and fluency can be assessed through perplexity scores and human evaluations (Wang et al., 2024). Additionally, the educational impact of the generated content can be measured through pre- and post-assessment scores, surveys, and feedback from students (Guo, 2022). Machine hallucination—where the model generates plausible but incorrect or irrelevant information—is a significant concern. Strategies to mitigate this issue include using high-quality training data, fine-tuning models to specific domains, and employing clear and specific prompts (Ji et al., 2023). Reducing hallucination is crucial for maintaining the integrity and reliability of AI-generated educational content.

Programming education often faces challenges such as meeting diverse learner needs, providing real-time feedback, and aligning learning content with curriculum goals. These issues are particularly critical for beginners who struggle with debugging, understanding abstract concepts, and applying theory to practice. The traditional learning approaches used

have low engagement due to abstract concepts, a steep learning curve, and a lack of timely feedback or individualized support. They often fail to accommodate diverse learning paces and styles, limiting students' ability to apply theoretical knowledge effectively. Large language models (LLMs) have been used to enhance personalized learning experiences. However, the quality of prompts pushed to these models remains a challenge. Poorly designed prompts result in irrelevant, misleading, or nonsensical responses. Some of these models also experience machine hallucination. This is where the model generates plausible but incorrect or irrelevant information. This study aimed to develop a framework to enhance personalized learning of computer programming using large language models. The framework utilizes prompt engineering to refine AI-generated content, ensuring its coherence, accuracy, and relevance. It reduces hallucinations through iterative feedback loops between students and instructors. The framework advances the use of LLMs in programming education, offering scalable, personalized, and adaptive solutions tailored to beginner programmers.

LITERATURE REVIEW

This study is grounded in constructivist learning theory, emphasizing active learner engagement, collaboration, and the co-construction of knowledge (Brown et al., 1989). Vygotsky's Zone of Proximal Development (ZPD) highlights the importance of scaffolding to help learners achieve tasks they cannot perform independently (Vygotsky, 1978). These theoretical principles align with personalized programming education, where beginner students require tailored guidance to overcome challenges in syntax, debugging, and logical structures (Mghsudi et al., 2021).

Generative AI tools, particularly LLMs, enable scaffolding by offering real-time feedback, customized exercises, and adaptive support (Naseer et al., 2024). However, for these models to align with educational principles, techniques like prompt engineering are critical (Mboya et al., 2025). Prompt engineering facilitates precision and contextual relevance, ensuring that AI-generated outputs support learner engagement and constructivist goals. Additionally, various AI techniques underpin this study, each offering unique benefits: Supervised learning is effective for predictable, structured tasks but limited in adaptability (Han et al., 2021); Reinforcement learning (RL) enables adaptive systems but presents challenges in defining reward functions and managing computational complexity (Chen et al., 2024); Transformer-based models (e.g., the GPT series) are powerful in generating human-like outputs but prone to hallucinations without refined prompts (Yenduri et al., 2024). This theoretical framework positions prompt engineering as a mechanism to operationalize these AI capabilities in programming education.

Large Language Models in Education

LLMs such as GPT-4 have demonstrated remarkable capabilities in generating human-like text, making them suitable for various applications, including education. Research has shown that these models can assist in providing personalized learning experiences by generating explanations, examples, and feedback tailored to individual students' needs (Wei

et al., 2021; Lin et al., 2021). However, the effectiveness of LLMs heavily depends on the quality of the prompts used and the techniques employed to train and fine-tune them.

Recent advancements in AI suggest that reinforcement learning offers promising opportunities to further improve LLMs for education. Unlike traditional supervised training on static datasets, where models learn from labeled examples, reinforcement learning allows models to optimize their behavior through feedback from an environment or user (Han et al., 2021). In educational contexts, such feedback could come from metrics like student engagement, task completion rates, or performance improvements. For example, Reinforcement Learning with Human Feedback (RLHF), a technique used to refine models like GPT-4, could be extended to prioritize outputs that maximize student comprehension and learning outcomes (Chen et al., 2021). Through iterative interactions, the model would learn to generate responses better aligned with pedagogical goals, reducing errors and hallucinations over time.

Furthermore, reinforcement learning could enhance the framework's adaptive learning algorithms. By dynamically adjusting task difficulty or feedback based on real-time student performance, RL-driven systems can create a more personalized and effective learning environment (Naser & Alavi, 2020). Studies have shown that RL algorithms can improve the decision-making of AI tutors, enabling them to provide targeted support for struggling students while appropriately challenging advanced learners (Zhou et al., 2022). Despite its potential, implementing RL in educational settings comes with challenges, such as defining meaningful reward functions, ensuring sufficient training data to simulate diverse learning scenarios, and managing the computational complexity of RL algorithms. Future research should explore these challenges and develop practical ways to integrate reinforcement learning into LLM-based educational frameworks.

Empirical studies have explored the use of generative AI for code generation in introductory programming courses. For instance, Finnie-Ansley *et al.* (2022), Hellas *et al.* (2023), and Kazemitabaar *et al.* (2023) demonstrated LLMs' capability to generate code and assist in debugging, highlighting their potential as virtual programming tutors. In programming education, such tools can provide real-time support as students learn to code, which is particularly valuable when students struggle with understanding syntax and logical structures (Chen et al., 2021). Additionally, LLMs have been integrated into intelligent tutoring systems (ITS) to provide adaptive learning experiences by analyzing student performance data and customizing learning pathways, thereby addressing individual learning needs (Figueiredo & García-Peñalvo, 2020; Kochmar et al., 2022). The use of LLMs in ITS has been shown to improve student engagement and learning outcomes, making them a promising tool for personalized education.

Despite their promise, LLMs have limitations. Bommasani *et al.* (2021) examined the opportunities and risks of these models, acknowledging their powerful content generation abilities but warning of issues like bias, misinformation, and content homogenization. A significant challenge is the tendency of LLMs to produce incorrect or irrelevant content (hallucinations), which can undermine the reliability of AI-generated educational material

(Cowan et al., 2023). Moreover, the quality of outputs is heavily dependent on prompt quality, underscoring the need for effective prompt engineering.

Several studies underscore these points. Finnie-Ansley *et al.* (2022) explored LLM use for code generation and debugging in introductory courses. While they found that LLM support can enhance learning outcomes, they did not address hallucinations or the alignment of AI outputs with curriculum objectives—factors essential for reliable educational use. Similarly, Figueiredo and García-Peñalvo (2020) integrated LLMs into an ITS for introductory programming. Their system improved student engagement and provided tailored support but lacked a detailed framework for leveraging prompt engineering to optimize AI outputs. These gaps highlight the need for systematic methods to reduce hallucinations, ensure content relevance, and enhance the adaptability of LLMs in diverse learning environments.

Prompt Engineering

Prompt engineering involves designing and refining prompts to guide LLMs in generating the desired output. The effectiveness of prompt engineering lies in its ability to enhance the accuracy, relevance, and coherence of LLM-generated content (Wang et al., 2024). Figure 1 below shows the basic large language model prompt cycle. Effective prompts are clear, specific, and contextually relevant.

Clarity ensures that the model understands the task, specificity helps narrow down the possible responses, and contextual relevance ensures the output is appropriate for the educational setting and aligns with learning objectives (Sivarajkumar et al., 2024). Additionally, instructive prompts provide clear guidance on what is expected in the response, further improving output quality (Cain, 2024). Recent research has explored how prompt structure influences LLM outputs. For instance, using multi-turn dialogue prompts or chaining simpler prompts can lead to more detailed and contextually accurate responses (Wang et al., 2024). This approach leverages the model's ability to maintain context over multiple interactions, resulting in outputs that better meet the specific needs of educational applications.

Various metrics can assess prompt performance. Precision and recall are commonly used to evaluate the accuracy and relevance of generated content. Perplexity scores assess the coherence and fluency of the text, while human evaluations (feedback from students and educators) provide insights into the educational impact of the prompts (Maity et al., 2024). These metrics are essential for continuously refining prompts and improving the overall effectiveness of LLM-based educational tools (Chen et al., 2021).

Machine Hallucination

Machine hallucination, where LLMs generate plausible but incorrect information, is a significant concern in educational applications. Various strategies have been proposed to mitigate this issue, including the use of high-quality training data, fine-tuning models to

specific domains, and employing clear, specific prompts (Ji et al., 2023). Ensuring that the model has access to accurate and relevant information during training can also help reduce the likelihood of hallucination.

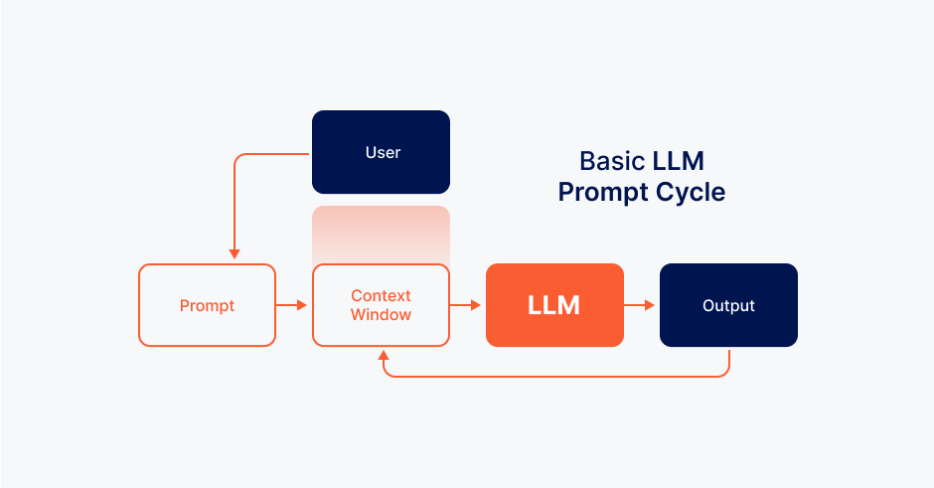

Figure 1. Basic LLM Prompt Cycle

Several studies have identified hallucination as a critical challenge in using LLMs for education. For example, Bommasani *et al.* (2021) highlighted that LLMs often generate plausible but factually incorrect outputs, particularly in domain-specific contexts such as programming. While they proposed fine-tuning as a mitigation strategy, they acknowledged that this approach alone cannot address the dynamic nature of real-time educational interactions. Similarly, Chen *et al.* (2023) observed that hallucination reduces user trust in AI systems, especially when generated outputs fail to align with curriculum objectives or contain logical errors. However, their work did not explore iterative feedback mechanisms as a potential solution.

Another effective strategy for reducing hallucination is incorporating external knowledge bases and verification mechanisms. By cross-referencing generated content with authoritative sources, LLMs can provide more accurate and reliable outputs (Taiwo et al., 2024). Furthermore, implementing user feedback loops—where students and educators can flag incorrect or misleading information—helps iteratively improve the model's performance.

Interactive techniques such as question-and-answer sessions and real-time feedback can also help mitigate hallucinations. These methods allow users to clarify ambiguities and correct errors on the spot, enhancing the accuracy and reliability of the generated content (Taiwo et al., 2024). Integrating such techniques into educational applications can create a more interactive and engaging learning experience, reducing the potential negative impact of hallucination.

Addressing Gaps in Research

The potential of LLMs in education is widely acknowledged, but significant gaps remain in their application to programming instruction, particularly in achieving personalized

learning outcomes (Chen et al., 2024). Programming courses often struggle to address the diverse needs of students due to limited adaptive content, one-size-fits-all teaching approaches, and insufficient real-time feedback mechanisms (MacNeil et al., 2023). These challenges disproportionately affect beginner programmers, who face difficulties such as debugging errors, understanding abstract programming concepts, and applying learned skills to practical problems.

Recent advancements in generative AI have introduced new opportunities to address these challenges. Studies have demonstrated that LLMs can generate personalized content, such as coding exercises and real-time feedback (Kochmar et al., 2022). However, existing applications often face critical issues, including:

1. Limited frameworks that integrate prompt engineering into active learning strategies for programming courses.
2. Hallucinations in AI-generated content can undermine trust and hinder learning outcomes.
3. Underexplored human-in-the-loop refinement, i.e., the role of students and educators in refining prompts and validating AI outputs.

While prompt engineering has emerged as a promising technique for improving the accuracy, coherence, and relevance of AI-generated content, its potential in personalizing programming education remains underexplored (Cowan et al., 2023). Current studies have yet to fully address how prompt engineering can be systematically applied to reduce hallucinations, align outputs with curriculum objectives, and support active learning. Additionally, gaps exist in leveraging iterative feedback mechanisms to refine prompts and ensure adaptability to learners' individual needs. This research aims to bridge these gaps by developing a comprehensive framework that integrates prompt engineering into programming education. The framework focuses on:

1. Enhancing the accuracy, coherence, and relevance of LLM-generated content.
2. Reducing hallucinations through iterative feedback loops involving students and educators.
3. Supporting active learning strategies that involve students in co-creating prompts and critiquing AI-generated outputs, fostering critical thinking and engagement.

In tackling these challenges, the proposed framework advances the use of LLMs in programming education, offering scalable, personalized, and adaptive solutions tailored to beginner programmers.

METHODOLOGY

This research adopts a systematic approach to develop, validate, and evaluate a framework for enhancing personalized learning in computer programming using LLMs. The methodology consists of five stages: framework design, data collection, experimental design, metrics for evaluation, and ethical consideration.

Framework Design

In the proposed framework, students engage with AI-generated content by interacting with code examples and explanations provided by the LLM. Active learning is promoted through coding exercises and problem-solving tasks. Students also play a role in providing feedback on the relevance and clarity of the generated content (Mollick et al., 2023). Lecturers, in turn, are responsible for designing and evaluating prompts, supplementing AI content with additional explanations, and monitoring student performance. This collaborative approach ensures that both students and educators contribute to the optimization of AI-driven learning experiences.

The framework was designed to address common challenges in programming education, such as limited personalized support and difficulties grasping abstract concepts. It incorporates LLM-generated content into active learning activities, emphasizing prompt engineering to optimize the quality of AI outputs. The framework's design includes tailored coding exercises, real-time feedback mechanisms, and a collaborative model involving both students and lecturers. Prompt engineering plays a central role, focusing on clarity, specificity, and contextual relevance to reduce hallucinations and improve content accuracy. Figure 2 illustrates the main components of the proposed framework.

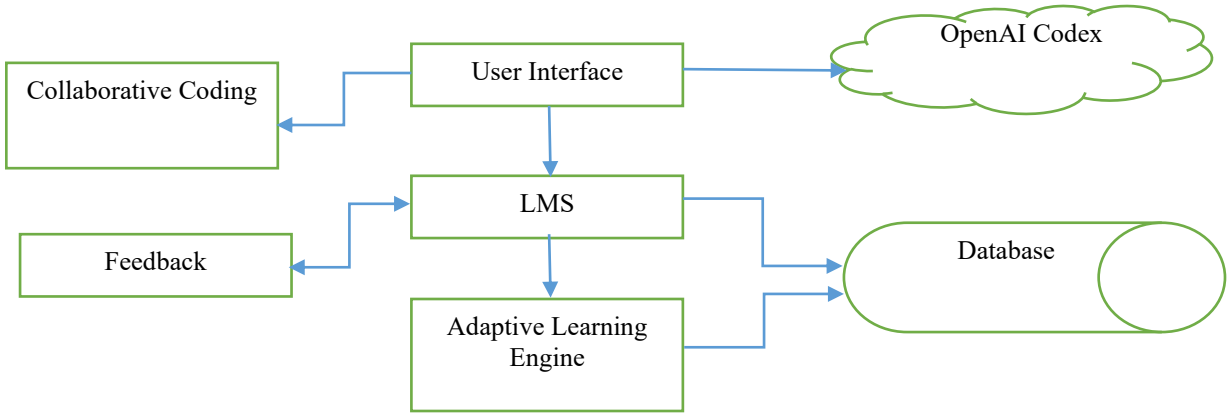

Figure 2. The Main Components of the Proposed Framework

In addition to these elements, the framework is built for seamless curriculum integration. It integrates active learning strategies to foster deeper student engagement. Students are empowered to co-create prompts, critique LLM-generated outputs, and participate in gamified learning experiences (e.g., prompt design challenges and debugging competitions) intended to motivate them and reinforce learning objectives.

The framework aligns with educational objectives by ensuring that AI-generated content effectively supplements existing teaching materials. This integration allows educators to adapt the framework to a wide range of programming topics while maintaining coherence with course structures and expected learning outcomes. LLM-generated exercises and explanations are embedded directly into the curriculum, so students can engage with personalized, contextually relevant materials without requiring significant changes to traditional instruction. In this way, the framework complements—rather than replaces—instructor-led teaching.

Adaptive learning algorithms tailor the difficulty of coding exercises and explanations to individual student needs. These algorithms use performance data (such as error patterns and completion times) to adjust the complexity of the materials presented. For example, students struggling with basic syntax might receive additional foundational exercises, while advanced learners are presented with more complex, logic-based problems (Zhai et al., 2021). This adaptability enhances the framework's ability to address diverse learner requirements and fosters a more inclusive learning environment (Kochmar et al., 2022).

To ensure practicality, the framework also prioritizes scalability. It is designed to be deployed across various institutional settings, accommodating both small and large class sizes. Scalability is achieved through modular components that can be tailored to specific class sizes, student demographics, and available resources. For instance, automated feedback mechanisms reduce reliance on instructor oversight, enabling large-scale implementation without compromising the quality of personalized learning experiences (Wei et al., 2021). Additionally, the integration of collaborative tools allows students to engage in peer learning activities, further enhancing the framework's suitability for diverse educational contexts.

The framework provides a structured method for integrating LLMs into programming instruction to improve individualized learning. By combining these interrelated elements, the framework establishes a flexible and dynamic educational setting that accommodates varied student needs and learning preferences, ultimately promoting improved learning outcomes and engagement.

Within the framework, the process of creating and using programming exercises follows an iterative life cycle involving instructors, students, and the LLM. Lecturers begin by creating prompts that the LLM uses to produce adaptive learning materials and exercises. Students then attempt these exercises and receive real-time feedback and hints from the LLM. Based on student performance and feedback, both students and lecturers refine the prompts and the generated content. This iterative feedback loop ensures continuous improvement of the materials and alignment with learning objectives. In this cycle, the LLM engages with both lecturers and students in real time, enabling dynamic interactions. Overall, the life cycle emphasizes that content creation and feedback are integrated into the learning process, with generative AI used for generating exercises and providing feedback on student attempts. Figure 3 provides a diagrammatic representation of this iterative process.

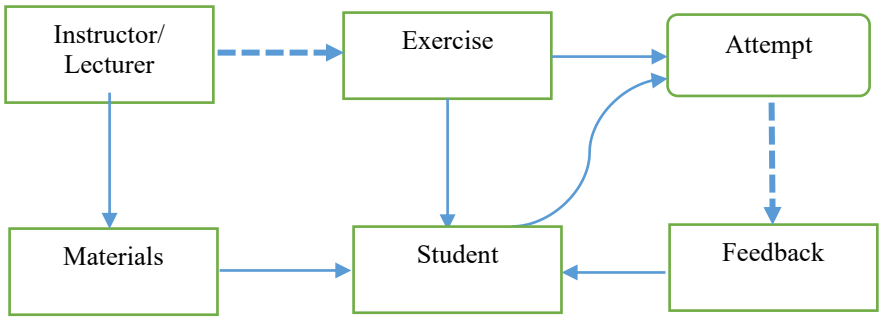

Figure 3. Lifecycle of Programming Exercises

Data Collection

Data were collected through an expert opinion survey involving stakeholders from multiple domains, namely: ICT industry experts, university lecturers, and advanced programming students. The survey explored the effectiveness of LLM-generated content, challenges associated with its use, and recommendations for improvement. Structured questionnaires were used to gather a mix of quantitative and qualitative insights. Advanced programming students provided feedback on the usability and educational value of AI-generated coding exercises, with emphasis on how well these exercises aligned with curriculum objectives and adapted to their skill levels. Lecturers evaluated the pedagogical relevance of the framework, focusing on how well the generated materials integrated into existing course structures and supported learning outcomes.

The framework's performance was assessed using a combination of quantitative and qualitative metrics. Quantitative measures included: (1) Accuracy – the proportion of correct AI-generated outputs, determined by comparing generated solutions and explanations against reference answers validated by programming instructors; (2) Relevance – the alignment of outputs with learning objectives, evaluated through Likert-scale feedback; (3) Coherence – measured by perplexity scores (with a threshold, e.g., perplexity < 30, indicating high fluency) and by Likert-scale feedback on logical flow and clarity. Qualitative evaluations were gathered through student and lecturer feedback, focusing on the perceived clarity, usefulness, and reliability of the outputs. The educational impact was further evaluated through pre- and post-assessment scores to measure improvements in students' programming proficiency.

Participant Observation

Participant observation was utilized as a complementary data collection method. The lecturer, serving as a participant observer, monitored students' interactions with the framework during classroom activities. This approach provided valuable real-time insights into student behavior, engagement levels, and challenges faced when using the framework. Observations focused on how students interacted with AI-generated exercises and feedback, the frequency and quality of student-initiated prompts or critiques of LLM-generated outputs, and engagement during gamified learning activities (such as debugging challenges or prompt design competitions). Observation data added qualitative context to the survey findings, capturing subtle behaviors not apparent from questionnaires alone. In particular, real-time observation helped illustrate how students approached problem-solving with the LLM's assistance and where they encountered difficulties, informing iterative improvements to the framework.

Participant Demographics

The survey involved participants in three groups: advanced programming students, university lecturers, and ICT industry professionals. Most participants had over two years of experience in programming or teaching programming-related courses, making them well-positioned to evaluate the framework's effectiveness. Figure 4 illustrates the demographic

distribution of the survey respondents. Students constituted about 60% of participants, reflecting the framework's primary focus on learners. Lecturers made up roughly 27%, contributing academic insights on pedagogy and curriculum alignment, while ICT professionals accounted for 13%, offering industry perspectives on practicality and scalability. This diverse representation helped ensure that the survey findings were well-rounded, incorporating input from multiple stakeholders in programming education.

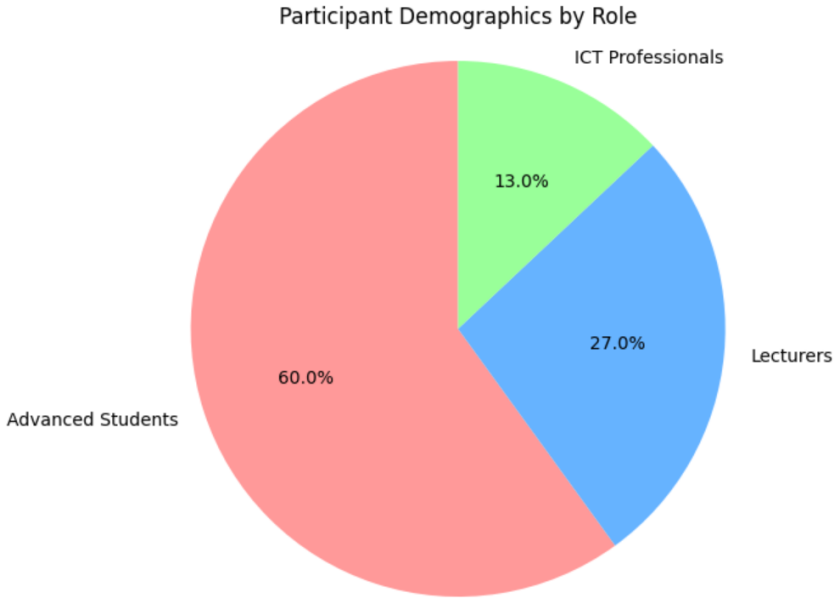

Figure 4. Opinion Survey Participant Demographics by Role

Experimental Design

The study employed a quasi-experimental design to compare traditional teaching methods with LLM-supported learning. The evaluation adopted a parallel instructional model in which students enrolled in an "Introduction to Computer Programming" unit of study were assigned to either a control group or an experimental group. The control group received instruction using traditional programming resources and materials, while the experimental group interacted with the developed LLM-based framework as a supplementary instructional tool. This arrangement was selected to isolate the framework's contribution by comparing learning outcomes across the two groups under similar instructional conditions. Data collection was carried out using structured pre- and post-tests consisting of 10 standardized questions, each worth 5 marks, to objectively measure students' understanding of core C programming concepts before and after engagement with the framework. The same set of questions was administered to both the experimental and control groups under identical conditions, allowing for a consistent comparison of knowledge gain attributable to the framework. This allowed for an objective assessment of knowledge gain attributable to the framework.

Two groups of first-year programming students participated: a control group that followed conventional classroom instruction and resources, and an experimental group that interacted with the LLM-supported framework (receiving personalized exercises, real-time feedback, and adaptive support). Both groups engaged with the same curriculum content. We conducted the study with 225 undergraduate programming students aged between 19

and 23 years at Murang'a University of Technology. A purposive sample of 125 students formed the experimental group and engaged with the GenAI-powered system over two weeks, while 100 students in the control group received conventional instruction. The gender distribution was 40% female and 60% male. Data collection involved structured questionnaires, open-ended surveys, and a pretest-posttest design aligned to core programming concepts. Performance was assessed through gain scores in pretest-posttest evaluations, mean ratings on personalization features, and learner feedback on usability and engagement.

All the participants completed a pre-assessment test to measure their baseline programming knowledge. After the intervention period, a post-assessment test was administered to evaluate learning gains. This pre/post comparison highlighted the framework's effectiveness. During the intervention, the experimental group engaged with the LLM framework in scheduled sessions, solving programming exercises and receiving AI-generated feedback and hints. They were also encouraged to critique the AI's outputs using structured rubrics. The control group followed traditional instructional methods, relying on textbooks and in-person guidance from the lecturer. They worked on the same exercises and received feedback directly from instructors or peers, without LLM support. The study ran for six weeks, with students in both groups participating in two sessions per week. This setup provided sufficient exposure for the experimental group to integrate the LLM into their learning routine and for measurable differences to emerge between the groups.

During the intervention, the lecturer observed both groups, documenting key aspects such as student engagement and participation rates, time spent on exercises and feedback interactions, and common errors or misconceptions that arose during programming tasks. These observations complemented the quantitative assessment data, helping to explain any differences observed between the control and experimental groups.

Before the experiment, both groups underwent pre-tests to establish their baseline knowledge and skills in programming. This was done using questionnaires and background assessment of the participants. Throughout the study, the experimental group interacted with the AI-powered framework, receiving personalized recommendations, feedback, and interactive tutoring based on their performance. The control group followed conventional instruction methods, without AI assistance. After the intervention period, both groups completed post-tests through quizzes to measure their learning progress and comprehension levels. Additionally, structured questionnaires and interviews were used to collect qualitative feedback from participants regarding their learning experiences. The structured questionnaire assessed variables such as usability, effectiveness, and engagement, while the interviews explored student perceptions, challenges, and areas for improvement.

Evaluation and Data Analysis

The experiment was conducted in an introductory programming course, focusing on first-year undergraduate students learning the C programming language. Students began by completing a pre-assessment test to establish their baseline knowledge. They then engaged

with their assigned learning approach: either interacting with the LLM-supported framework or following traditional methods. Throughout the process, students and lecturers provided iterative feedback on the AI-generated content to refine the prompts and outputs. At the end of the study, a post-assessment test measured learning gains, which were compared to the baseline scores.

To evaluate the effectiveness of curriculum integration, the framework's outputs (exercises and explanations) were reviewed for alignment with course objectives. Lecturers assessed this alignment by rating the contextual relevance of LLM-generated content on a Likert scale. Additionally, student engagement metrics (such as time spent on exercises and participation rates) were analyzed to determine whether the framework enhanced engagement relative to the traditional approach.

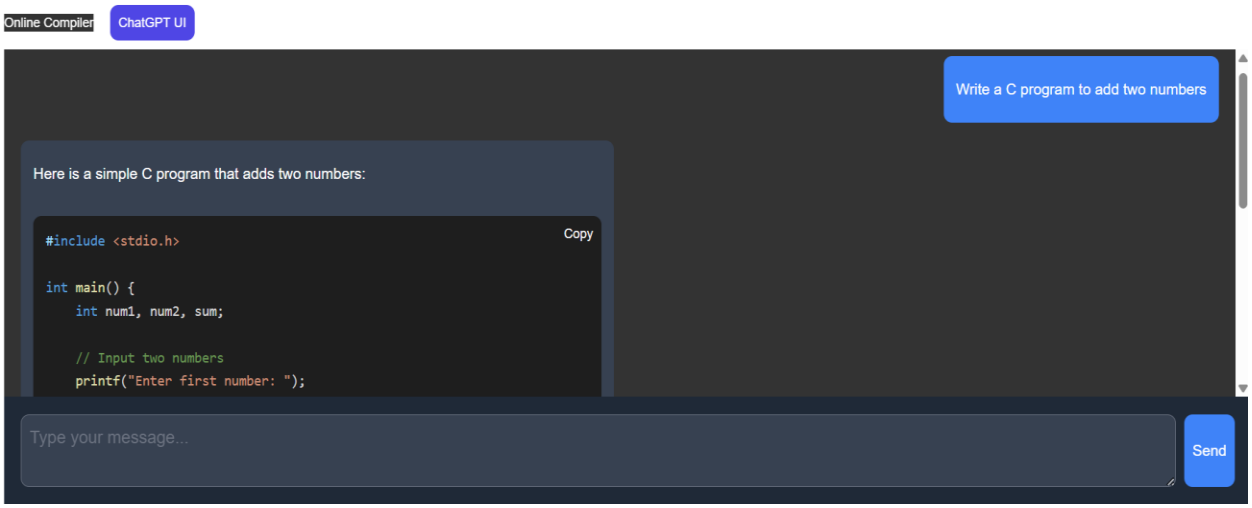

Figure 5: Prompt on the Integrated LLM

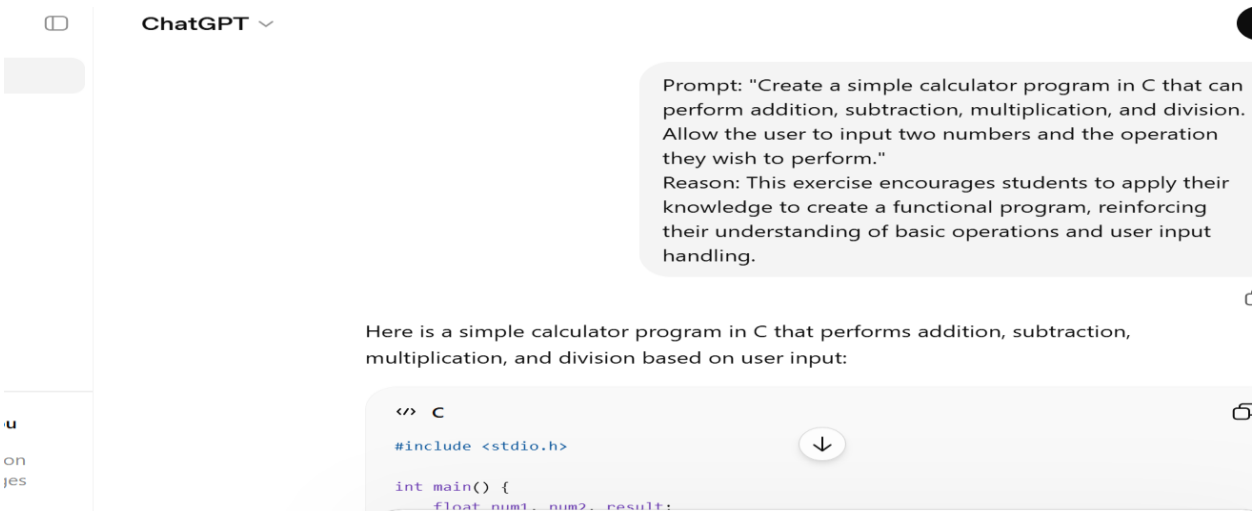

Figure 6. Prompt on ChatGPT

We used adaptive learning algorithms, which were evaluated by tracking changes in student performance over time. Metrics such as reduced error rates and the ability to progress to more advanced exercises were considered. We also gathered student feedback

on task difficulty to verify that the adaptive system appropriately adjusted to individual needs. To strengthen the validity of the results, we performed a comparative analysis between the control and experimental groups. Both groups' pre- and post-assessment scores were compared to measure learning gains.

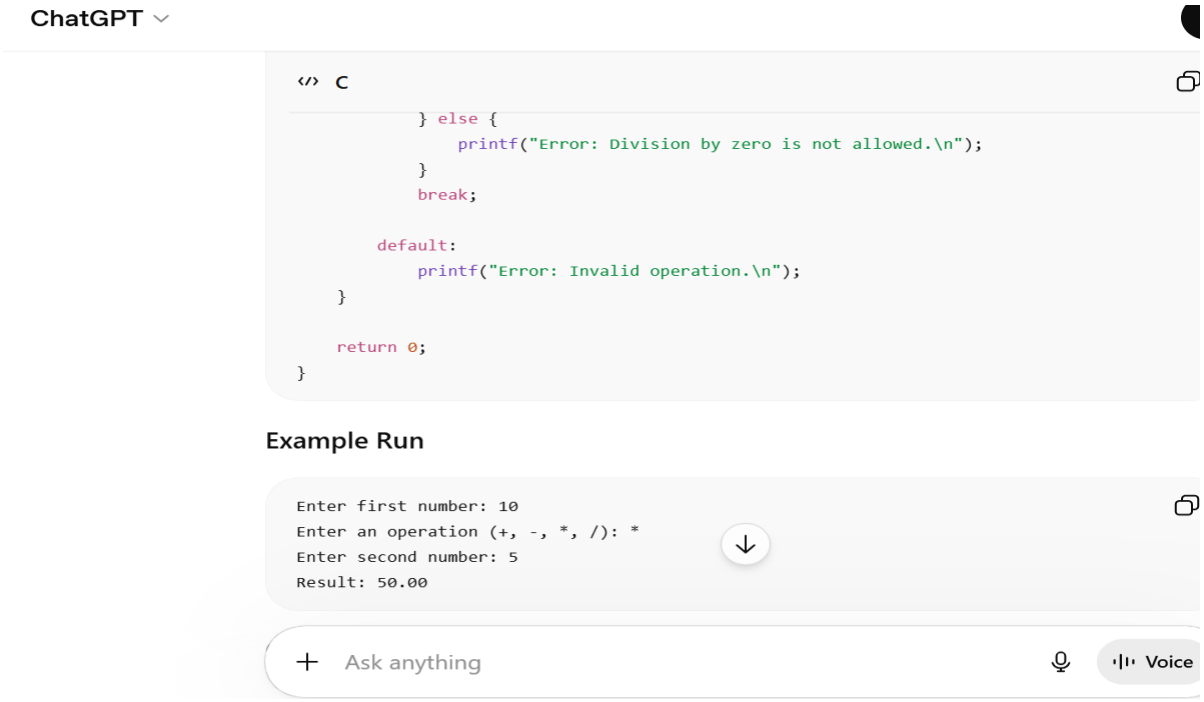

Figure 7. Follow-Up prompt on ChatGPT

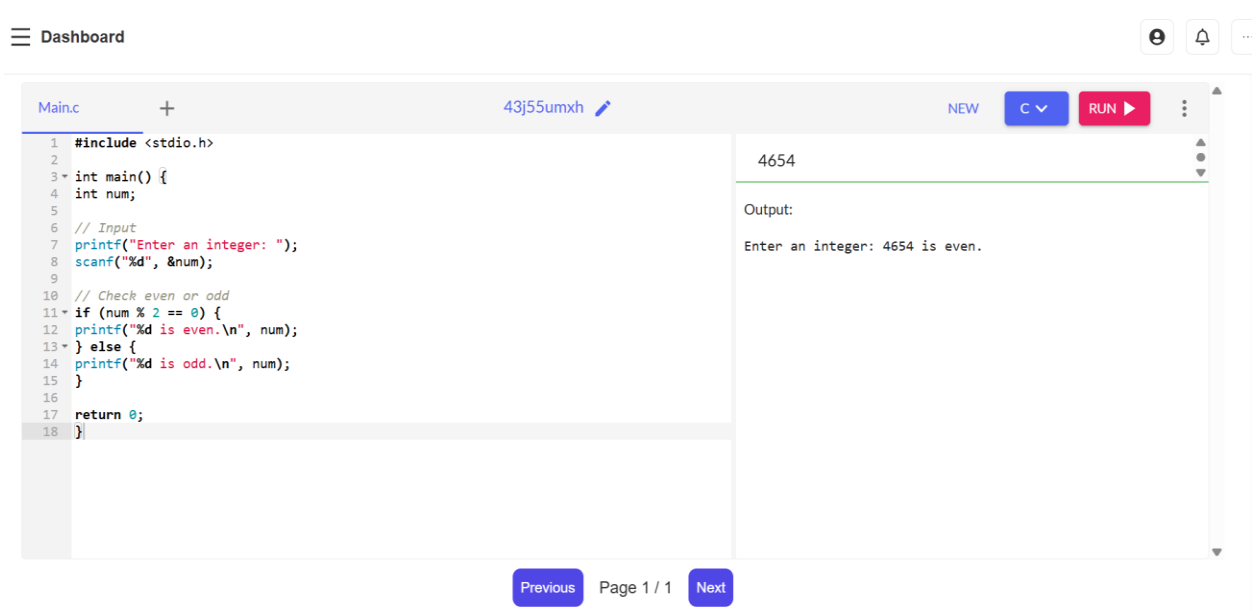

Figure 8. Compiler integrated in the Learning Management System

Engagement metrics (such as average time spent on tasks, participation frequency, and task completion rates) were analyzed to see how each approach influenced student involvement in learning activities. Additionally, we monitored error reduction rates to

determine how effectively each group resolved common coding mistakes during exercises. Finally, we evaluated the quality of feedback through student surveys, asking participants in both groups to rate the clarity and usefulness of the feedback they received (AI-driven feedback for the experimental group vs. human feedback for the control group). This comparative analysis provided insights into the added value of LLM-supported learning over traditional approaches, ensuring the framework's effectiveness was thoroughly evaluated. Figures 4, 6, 7, and 8 below show sample screenshots of how the participants were interacting with the LLMs and the compiler integrated with the learning management system.

Ethical Considerations

Integrating LLMs into an educational framework introduces important ethical considerations, particularly around data privacy and bias mitigation. Addressing these issues is crucial for the responsible use of AI technologies and for maintaining trust among students, instructors, and other stakeholders. The framework incorporates specific strategies to uphold ethical integrity without compromising functionality. These include:

i. **Data Privacy:** Protecting the privacy of students and lecturers is fundamental. To comply with institutional policies and regulations like the General Data Protection Regulation (GDPR), the framework implements several measures. First, all user data (including code submissions and interaction logs) are anonymized by removing identifiable information before processing. This ensures that personal details cannot be traced back to individuals, safeguarding their privacy. Additionally, secure data storage and transmission are prioritized. The framework uses robust encryption protocols to protect data both in transit and at rest. All communication between students, lecturers, and the framework is encrypted to prevent unauthorized access or data breaches. Furthermore, participants are provided with detailed consent forms explaining what data will be collected, how it will be stored, and how it will be used. This transparency allows students and lecturers to make informed decisions about their participation and includes an option to opt out if they have concerns.

ii. **Bias Mitigation:** Bias in AI-generated content can perpetuate stereotypes and lead to inequitable learning experiences. The framework employs multiple strategies to mitigate bias. First, the LLMs are fine-tuned on datasets that are diverse and representative of various cultural, linguistic, and socioeconomic backgrounds. By training on a broad range of data, the likelihood of biased outputs is reduced, and the generated content becomes more inclusive and relevant to a wide spectrum of learners. A bias-detection module is also integrated into the framework to flag potentially biased or sensitive outputs. Flagged content is then reviewed by instructors (and even by students during critique exercises) to determine if it contains bias or inaccuracies. If biased content is identified, it can be corrected or removed, and the incident is used as a learning opportunity to improve the system's fairness. Moreover, feedback loops are established for continuous refinement. Feedback from users regarding any biased or inappropriate outputs is analyzed to adjust the LLM's parameters or training data, thereby iteratively improving the model's performance with respect to fairness.

iii. **Preventing Over-Reliance on AI:** A potential risk of introducing LLMs into education is that students might become overly reliant on AI-generated solutions, which could impede their development of independent problem-solving skills. The framework addresses this by embedding safeguards to encourage critical engagement rather than passive acceptance of AI output. First, as noted, students are actively involved in critiquing AI-generated content and co-creating prompts, which requires them to evaluate and think deeply about the material rather than using it uncritically. These activities promote critical thinking and a better understanding of programming concepts. The framework also uses a phased guidance approach: the level of AI assistance is gradually reduced as students gain proficiency. Early in a course or unit, exercises may come with substantial AI hints and guidance to help novices. As students improve, the AI provides less detailed help, prompting students to rely more on their own knowledge and problem-solving abilities. For example, initial exercises might highlight syntax errors and suggest fixes, whereas later exercises might only indicate that an error exists, leaving the student to diagnose and fix it. Additionally, instructors intentionally design certain assignments or tests where AI assistance is limited or disallowed, ensuring that students practice solving problems without the AI's help. Peer collaboration is also encouraged for tasks without AI intervention, further reinforcing independent learning and group problem-solving skills.

iv. **Balancing Ethical Considerations with Functionality:** While data privacy and bias mitigation are priorities, these measures are balanced with the framework's need to function effectively as a learning tool. For instance, data anonymization is implemented in a way that preserves the information needed for personalization; the system might track an anonymized student ID's progress to adapt the difficulty of exercises, without storing the student's name. Similarly, bias monitoring and content review processes are designed to be efficient so that they do not interrupt the real-time feedback loop that students rely on during practice sessions. By embedding ethical considerations into the framework's design from the outset, the system maintains both ethical integrity and practical usability. In essence, the framework strives to protect privacy and ensure fairness while still delivering timely, adaptive support to students as they learn programming.

RESULTS

To inform the framework's development, we first gathered expert opinions via a structured survey. Participants included advanced programming students, university lecturers, and ICT industry professionals, each providing a unique perspective on the framework's utility and challenges. Quantitative data were collected using five-point Likert scale items (1 = Strongly Disagree, 5 = Strongly Agree) focusing on the effectiveness of prompt engineering, the educational usefulness of the AI outputs, and potential issues with AI-generated content. Qualitative feedback was also gathered through open-ended questions. The key findings are organized into themes corresponding to the evaluation criteria.

Figure 5 shows the summary of results obtained from experiments with the proposed framework of this study. It summarizes the average ratings for each evaluation theme (clarity, accuracy, feedback/collaboration, practicality, and readiness). The results are discussed below.

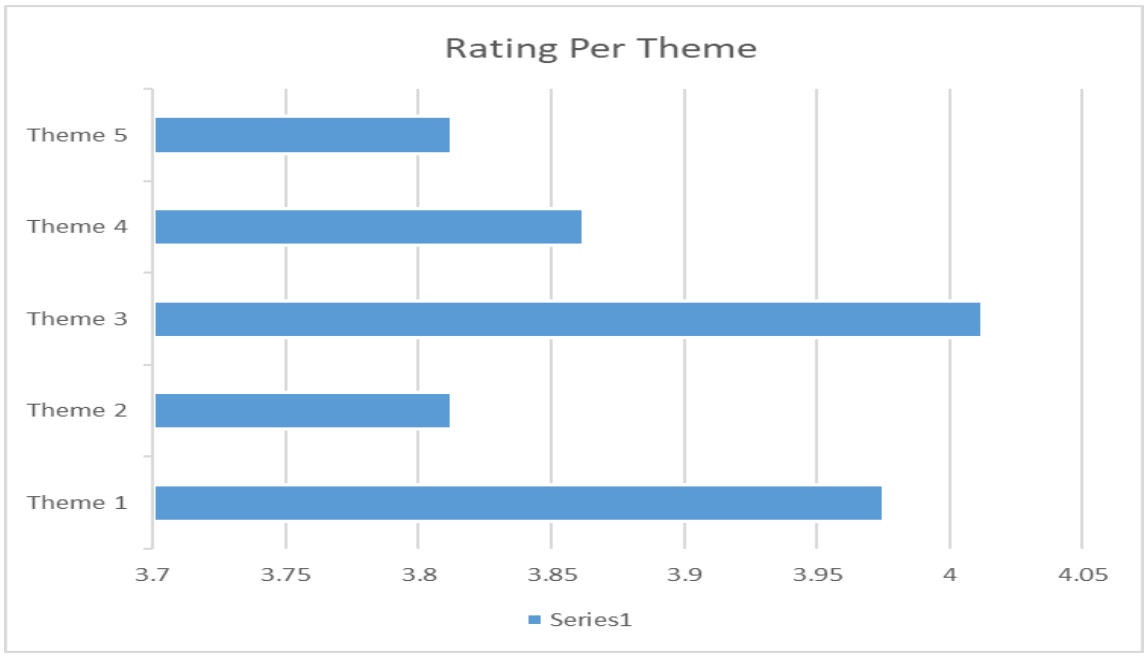

Figure 6: Average rating per theme

Table 1. Mann-Whitney U Test Results

Test	U	Z	p-value (2-tailed)	Median (Control)	Median (Experimental)	Interpretation
Post-test Score Comparison	1,725.5	-2.634	0.008	63	67	Statistically significant difference ($p < .01$)

To assess the effectiveness of the GenAI-powered learning framework in improving programming competencies, a pretest-posttest evaluation was conducted comparing two groups: the experimental group (n = 125), which used the developed framework, and a control group (n = 100), which received conventional instruction without GenAI support. As summarized in Table 1, the experimental group recorded a pretest mean of 57.16 and improved to a posttest mean of 66.24, yielding a gain of +9.08 marks. The control group began with a higher pretest mean of 60.94 and achieved a posttest mean of 63.08, resulting in a comparatively lower gain of +2.14 marks. To determine whether the observed performance difference between the groups was statistically significant, a Mann-Whitney U test was conducted on the posttest scores. The test yielded a U-value of 1,725.5, a Z-score of -2.634, and a p-value of 0.008, indicating that the difference between the experimental and control groups was statistically significant at the $p < 0.01$ level. Table 4.9 illustrates the Mann-Whitney U Test results of the posttest score.

DISCUSSION

The key findings, organized into themes corresponding to the evaluation criteria and other results, are discussed herein.

Clarity: The clarity of AI-generated explanations is a fundamental metric for assessing the framework's potential to enhance personalized learning (Mboya et al., 2025). Participants rated the LLM (ChatGPT-4 in this study) on its ability to deliver step-by-step guidance, use appropriate terminology, and explain programming concepts effectively. The average score for clarity-related items was 4.13, indicating that the explanations were generally perceived as clear and well-structured. In particular, the use of relevant terminology scored 4.06, and the quality of step-by-step explanations was rated 4.31. These findings suggest that prompt engineering contributed to high clarity by eliciting structured, logical, and student-friendly explanations from the LLM. Such clarity is crucial, as it directly supports better learner comprehension and engagement with programming tasks (Cain, 2024; Taiwo et al., 2024).

Accuracy of Explanations: Accuracy is critical in programming education, where incorrect or ambiguous information can hinder learning. This theme assessed the correctness, logical consistency, and quality of examples provided by the LLM. The average score for accuracy was 4.38. Notably, the highest rating (4.44) was for the correctness of code examples and explanations, indicating that participants found the LLM's outputs largely accurate and on point. Logical consistency was also well-rated, suggesting that the explanations generally made sense and followed sound reasoning. However, explanations for edge cases scored lower, at 3.38. This indicates that the framework requires further refinement to handle complex or less common scenarios in programming. These results align with the research objective of reducing hallucinations and ensuring reliability in AI outputs. The current results on the relevance of accuracy and reliability of information/ output in LLM models also confirm the previous findings (Wang et al., 2024; Taiwo et al., 2024). Overall, the high accuracy scores build trust in the LLM as a learning tool, but the lower score for edge cases underscores the need to refine prompt engineering techniques to cover more complex programming problems and improve the depth of AI-generated explanations.

Opportunities for Feedback and Collaboration: Interactive learning and collaboration are key to fostering engagement in programming education. This theme evaluated how well the framework supports feedback, peer interaction, and collaborative problem-solving among learners. The average score for this theme was 3.69, indicating moderate effectiveness. Participants gave high marks for the timeliness of feedback, with a score of 4.13, suggesting that the framework's real-time hints and corrections were valued. On the other hand, the framework's ability to stimulate peer-to-peer collaboration and discussion was rated lower (around 3.68). These findings align with previous studies that indicated adequate feedback is an important aspect in enhancing learning in programming education (Naseer et al., 2024; Zhou et al., 2022). The current findings further reveal an area for improvement: while the AI-based system excels at providing individualized feedback, it could better facilitate collaborative learning experiences. In practice, this might mean

incorporating more features that encourage students to discuss AI outputs with each other or work together on challenges within the framework. Addressing this gap could enhance the framework's overall impact by adding a social learning dimension to its personalized support.

Practicality and Uniqueness of Content: Programming exercises need to be practical and varied to maintain student interest and relevance to real-world applications. This theme assessed whether the content generated by the LLM met those criteria. The average score was 3.94. Participants rated the practicality of exercises for real-world applications at 4.06, indicating that the AI-generated tasks were largely seen as useful and applicable to practical programming contexts. The uniqueness (novelty) of the exercises was rated slightly lower, at 3.75, suggesting that some participants felt the exercises were somewhat repetitive or not particularly innovative. These results imply that while the framework is effective at producing exercises that align well with curriculum goals and real-world needs, there is room to increase the diversity and creativity of the problems presented. As observed in previous studies, ensuring a wider variety of exercise types or scenarios enhances learners' engagement and challenges them in different ways, thus enhancing adaptability and preventing monotony in the learning experience (MacNeil et al., 2023; Wei et al., 2021).

Readiness for Classroom Use: This theme explored whether the framework's outputs were ready for immediate use in a classroom setting without significant modification. The average score was 3.88, reflecting moderate readiness. The alignment of exercises with introductory programming learning objectives received a strong score of 4.13, demonstrating that the content fits well with typical course goals. However, the need for modifications before classroom use was rated at 3.12. In practical terms, this means that instructors saw value in the AI-generated content but often anticipated having to tweak or curate it before deploying it directly in their classes. For example, an instructor might want to edit an AI-generated problem description for clarity or adjust difficulty levels. To improve in this area, future iterations of the framework should aim to produce content that requires minimal adjustments—essentially "plug-and-play" exercises and explanations that instructors can trust out of the box. Literature underscores the need for domain-specific customization of AI- generated content to enhance contextual relevance and usefulness to learners with diverse needs (Sivarajkumar et al., 2024; Wang et al., 2024).

The empirical results of this study offer compelling evidence regarding the practical impact of the developed LLM-based framework on student engagement, content understanding, and the overall learning experience in programming education. In summary, the expert survey findings highlight the potential of the proposed LLM-supported framework in a practical sense to enhance personalized learning in programming education. The results demonstrate that prompt engineering plays a pivotal role in ensuring clarity and accuracy when handling practical programming tasks. Participants generally found the AI-generated explanations to be clear and correct, confirming one of the study's core hypotheses. This confirms previous research that underscores accuracy as a fundamental element of effective AI frameworks in programming education (Mboya et al., 2025).

The framework also showed strength in providing timely, individualized feedback, addressing a common challenge in programming instruction (namely, the lag in feedback that students often experience). The theoretical results of this study underscore the importance of design-driven personalization in AI-powered learning systems. They suggest that effective GenAI frameworks for programming education must go beyond novelty and automation, and instead embed educationally grounded principles that support the learner's development through intelligent, responsive, and context-aware interaction. This theoretical foundation not only informed the design of the current framework but also offers a replicable model for future research and system development in AI-supported personalized learning.

Previous studies have indicated that timely feedback is an important component in supporting students' meaningful and sustained engagement (Cheah, 2020; Chen et al., 2024). Moreover, the evaluation of the developed framework pointed out areas for improvement, which are part of the key findings in the current study. These improvements include enhancing the collaborative features that could foster peer learning and discussion around the AI-generated content. Additionally, the current findings reveal the need for increasing the novelty and variety of exercises, which could keep students more engaged and exposed to a broader range of problem-solving situations. The current findings also indicate that reducing the need for instructor intervention by improving the polish of AI outputs would make the framework more immediately useful in real classrooms.

The results from the learning outcomes comparison through a pretest-posttest design demonstrate the practical value of the GenAI-based framework in enhancing student learning outcomes. Despite beginning with a lower average performance, the experimental group surpassed the control group in posttest scores. This outcome highlights the framework's effectiveness in providing adaptive feedback, structured task sequencing, and context-aware guidance that support learner progress, particularly for those who may start with weaker programming skills. The statistically significant results validate the framework's capacity to deliver personalized, impactful learning experiences in programming education.

CONCLUSIONS AND RECOMMENDATIONS

This paper presented a comprehensive framework for enhancing personalized learning in computer programming education through the integration of LLMs. At the core of the framework lies prompt engineering, which is a critical tool for improving the clarity, accuracy, and relevance of AI-generated content. Additionally, the study highlighted the importance of careful curriculum integration, adaptive learning algorithms, and scalability in addressing the diverse needs of learners and ensuring the framework's practical applicability in real-world educational settings.

The proposed framework introduces effective strategies to design prompts, reduce hallucinations, and align generated content with curriculum objectives. By incorporating real-time feedback mechanisms and active learning approaches (such as prompt co-creation and critique), the framework fosters deeper engagement and supports individualized

learning experiences. Results from the expert evaluation affirm the framework's potential to deliver scalable, adaptive, and student-centered support in programming education.

While this work lays a strong foundation, future research should focus on extending the framework's implementation and evaluation in live classroom settings. This will include conducting controlled studies to measure its impact on student learning outcomes, refining collaborative features to enhance peer interaction, and incorporating additional AI techniques (such as reinforcement learning) to further optimize educational content. Additionally, the framework's adaptability to diverse programming courses and its scalability across different educational contexts can be explored to ensure broader applicability.

IMPLICATIONS

The empirical results of this study offer compelling evidence regarding the practical impact of the developed LLM-based framework on student engagement, content understanding, and the overall learning experience in programming education. In summary, the expert survey findings highlight the potential of the proposed LLM-supported framework in a practical sense to enhance personalized learning in programming education. The results demonstrate that prompt engineering plays a pivotal role in ensuring clarity and accuracy when handling practical programming tasks. Participants generally found the AI-generated explanations to be clear and correct, confirming one of the study's core hypotheses. This confirms previous research that underscores accuracy as a fundamental element of effective AI frameworks in programming education (Mboya et al., 2025).

The framework also showed strength in providing timely, individualized feedback, addressing a common challenge in programming instruction (namely, the lag in feedback that students often experience). The theoretical results of this study underscore the importance of design-driven personalization in AI-powered learning systems. They suggest that effective GenAI frameworks for programming education must go beyond novelty and automation, and instead embed educationally grounded principles that support the learner's development through intelligent, responsive, and context-aware interaction. This theoretical foundation not only informed the design of the current framework but also offers a replicable model for future research and system development in AI-supported personalized learning.

ACKNOWLEDGEMENT

We thank the co-authors for their contributions to this work.

FUNDING

Kenya Education Network (KENET) funded the research.

DECLARATIONS

Conflict of Interest

The researcher declares no conflict of interest in this study.

Informed Consent

Each of the authors has participated in the work.

Ethics Approval

The work has been approved by the relevant ethics body in Kenya.

REFERENCES

Banjade, R., Oli, P., Sajib, M. I., & Rus, V. (2024). Identifying gaps in students' explanations of code using LLMs. In A. M. Olney, I.-A. Chounta, Z. Liu, O. C. Santos, & I. I. Bittencourt (Eds.), *Artificial Intelligence in Education* (Lecture Notes in Computer Science, Vol. 14830, pp. 268–275). Cham: Springer. https://doi.org/10.1007/978-3-031-64299-9_21.

Bommasani, R., Drew, A., & Ehsan, A. (2021). On the opportunities and risks of foundation models. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2108.07258.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*(1), 32-42. https://doi.org/10.3102/0013189x018001032

Cain, W. (2024). Prompting change: Exploring prompt engineering in large language model AI and its potential to transform education. *TechTrends, 68*(1), 47–57. https://doi.org/10.1007/s11528-023-00896-0.

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology, 12*(2), ep272. https://doi.org/10.30935/cedtech/8247.

Chen, J., Chen, J., Liu, Z., & Huang, X. (2024). When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web, 27*(4), 42. https://doi.org/10.1007/s11280-024-01276-1.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, O., Kaplan, J., & Zaremba, W. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Chen, Y., Fu, Q., Yuan, Y., Wen, Z., Fan, G., Liu, D., & Xiao., Y. (2023). Hallucination detection: Robustly discerning reliable answers in large language models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (pp. 245-255). https://doi.org/10.1145/3583780.3614905

Cowan, B., Watanobe, Y., & Shirafuji, A. (2023). Enhancing programming learning with LLMs: Prompt engineering and flipped interaction. In *Proceedings of the 2023 4th Asia Service Sciences and Software Engineering Conference* (pp. 10-16). https://doi.org/10.1145/3634814.3634816

Durak, H. Y. (2020). Modeling different variables in learning basic concepts of programming in flipped classrooms. *Journal of Educational Computing Research, 58*(1), 160–199. https://doi.org/10.1177/0735633119827956.

Figueiredo, J., & García-Peñalvo, F. J. (2020). Intelligent tutoring systems approach to introductory programming courses. In *Proceedings of the Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 34–39). New York, NY: ACM. https://doi.org/10.1145/3434780.3436614.

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference* (pp. 10–19). New York, NY: ACM. https://doi.org/10.1145/3511861.3511863.

Guo, L. (2022). Using metacognitive prompts to enhance self-regulated learning and learning outcomes: A meta-analysis of experimental studies in computer-based learning environments. *Journal of Computer Assisted Learning, 38*(3), 811–832. https://doi.org/10.1111/jcal.12650.

Han, J. M., Igor, B., Tao, X., & Ray, A. (2021). Unsupervised neural machine translation with generative language models only. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2110.05448.

Ji, Z., Su, D., Xu, Y., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55*(12), 1–38. https://doi.org/10.1145/3571730.

Kochmar, E., Vu, D., Belfer, R., Gupta, V., Serban, V., & Pineau, J. (2022). Automated data-driven generation of personalized pedagogical interventions in intelligent tutoring systems. *International Journal of Artificial Intelligence in Education, 32*(2), 323–349. https://doi.org/10.1007/s40593-021-00267-x.

Li, Q., Fu, L., Weiming, Z., Xianyu, C., Jingwei, Y., Wei, X., Weinan, Z., Ruiming, T., & Yong, Y. (2024). Adapting large language models for education: Foundational capabilities, potentials, and challenges. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2401.08664.

Lin, V., Wang, T., Vishrav, C., Jeff, W., Diab, M., & Li, Z. (2021). Few-shot learning with multilingual language models. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2112.10668.

MacNeil, S., Tran, A., Arto, H., Kim, J., Sami, S., Denny, P., Bernstein, S., & Juho, L. (2023). Experiences from using code explanations generated by large language models in a web software development e-book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 931–937). New York, NY: ACM. https://doi.org/10.1145/3545945.3569785.

Maghsudi, S., Lan, A., Xu, J., & Van Der Schaar, M. (2021). Personalized education in the artificial intelligence era: What to expect next. *IEEE Signal Processing Magazine, 38*(3), 37–50. https://doi.org/10.1109/MSP.2021.3055032.

Maity, S., Deroy, A., & Sarkar, S. (2024). Exploring the capabilities of prompted large language models in educational and assessment applications. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2405.11579.

Marvin, G., Hellen, N., Jjingo, D., & Nakatumba-Nabende, J. (2024). Prompt engineering in large language models. In I. J. Jacob, S. Piramuthu, & P. Falkowski-Gilski (Eds.), *Data Intelligence and Cognitive Informatics* (Algorithms for Intelligent Systems, pp. 387–402). Singapore: Springer. https://doi.org/10.1007/978-981-99-7962-2_30.

Mollick, E. R., & Mollick, L. (2023). Using AI to implement effective teaching strategies in classrooms: Five strategies, including prompts. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4391243.

Naseer, F., Khalid, M. U., Ayub, N., Rasool, A., Abbas, T., & Afzal, M. W. (2024). Automated assessment and feedback in higher education using generative AI. In R. C. Sharma & A. Bozkurt (Eds.), *Advances in Educational Technologies and Instructional Design* (pp. 433–461). Hershey, PA: IGI Global. https://doi.org/10.4018/979-8-3693-1351-0.ch021.

Naser, M. Z., & Alavi, A. (2020). Insights into performance fitness and error metrics for machine learning. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2006.00887.

Patil, R., Heston, T. F., & Bhuse, V. (2024). Prompt engineering in healthcare. *Electronics, 13*(15), 2961. https://doi.org/10.3390/electronics13152961.

Sivarajkumar, S., Kelley, M., Samolyk-Mazzanti, A., Visweswaran, S., & Wang, Y. (2024). An empirical evaluation of prompting strategies for large language models in zero-shot clinical natural language processing: algorithm development and validation study. *JMIR Medical Informatics*, *12*, e55318. DOI: 10.2196/55318.

Szymanski, A., Wimer, B. L., Anuyah, O., Eicher-Miller, H. A., & Metoyer, R. A. (2024). Integrating expertise in LLMs: crafting a customized nutrition assistant with refined template instructions. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (pp. 1-22). https://doi.org/10.1145/3613904.3641924

Taiwo, R., Idris, T., Bello, Fatoama, S., Abdulai, A., Yussif, B., Abiodun, S., Saka, A., & Zayed, T. (2024). Generative AI in the construction industry: A state-of-the-art analysis. *arXiv preprint.* https://doi.org/10.48550/ARXIV.2402. 09939.

Vygotsky, L. S. (Ed.). (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.

Wang, L., Xi, C., Deng, X., Wen, H., You, M., Liu, W., Li, Q., & Jian, L. (2024). Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs. *NPJ Digital Medicine, 7*(1), 41. https://doi.org/10.1038/s41746-024-01029-4.

Wang, T., Zhou, N., & Chen, Z. (2024). Enhancing computer programming education with LLMs: A study on effective prompt engineering for Python code generation. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2407.05437.

Wei, J., Bosma, Y., Zhao, G., Wei, Y., Lester, B., Nan, D., Dai, M., & Quoc, L. (2021). Fine-tuned language models are zero-shot learners. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2109.01652.

Wood, D., & Moss, H. (2024). Evaluating the impact of students' generative AI use in educational contexts. *Journal of Research in Innovative Teaching & Learning, 17*(2), 152–167. https://doi.org/10.1108/JRIT-06-2024-0151.

Yenduri, G., Ramalingam, M., Selvi, G.C., Supriya, Y., Srivastava, G., & Maddikunta, R. (2024). GPT (Generative Pre-Trained Transformer)—A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access, 12*, 54608–54649. https://doi.org/10.1109/ACCESS.2024.3389497.

Zhai, X., Chu, X., Chai, C. S., & Wang, C. (2021). A review of artificial intelligence (AI) in education from 2010 to 2020. *Complexity, 2021*, Article 8812542, 1–18. https://doi.org/10.1155/2021/8812542.

Zhou, W., Wu, W., Chen, M., Liu, J., Xiao, X., & Wu, H. (2022). Faithfulness in natural language generation: A systematic survey of analysis, evaluation, and optimization methods. *arXiv preprint*. https://doi.org/10.48550/ARXIV.2203.05227.

Authors' Biographies

Dr. Erick Omuya is a researcher and AI-Machine Learning lecturer at Machakos University, Kenya. Dr. Omuya holds a PhD in Information Technology from the Jomo Kenyatta University of Agriculture and Technology (JKUAT), Kenya. His current research and publications deal with developing AI models for improving the performance of classification, clustering, and prediction using Machine/deep learning algorithms and Natural Language Processing as applied in sentiment analysis, market segmentation, among others.

Dr. Geoffrey Mariga is a Senior Lecturer in Information Technology at Murang'a University of Technology, Kenya. He holds a PhD in Information Technology - Machine Learning from Jomo Kenyatta University of Agriculture and Technology (JKUAT). His research focuses on machine learning, deep learning, data science, and natural language processing, among others. He has supervised numerous master's and doctoral students, led curriculum development initiatives, and published extensively in reputable journals.

Dr. Faith Musyoka is a Senior Lecturer in the Computing and Information Technology Department at the University of Embu, Kenya. She holds a PhD in Information Technology from Kabarak University, Kenya. Her research interests span Artificial Intelligence, Responsible AI, Internet of Things (IoT), and digital transformation. She has published articles in high-impact peer-reviewed journals, and her current research focuses on developing trustworthy AI systems, AI governance frameworks, and digital health solutions.

Dr. Joyce W. Gikandi is a specialist in Information Systems and Educational Technology. She is currently a Senior Lecturer at Mount Kenya University (MKU). Her current research interests include E-Learning developments, ICT application in social and health sciences, AI Integration in education for adaptive learning, and digital skills for youth and women empowerment. Another area of interest is promoting innovative adaptation and use of open-source software and content for educational inclusivity.

Mr. Fredrick Muema Mboya is a researcher and Assistant Lecturer at Zetech University, Kenya. He holds a Master of Science in Information Technology from Murang'a University of Technology, Kenya. His research interests include Artificial Intelligence, Machine Learning, and Natural Language Processing, with a focus on intelligent systems and data-driven solutions.