Artificial Intelligence in Education

33

What do students want from AI? Exploring expectations and preferences in programming education

Mubina Kamberovic and Senka Krivic
Faculty of Electrical Engineering, University of Sarajevo, Sarajevo, Bosnia and Herzegovina

Received 31 August 2025
Revised 18 November 2025
Accepted 31 December 2025

Abstract

Purpose – This study explores first-year Electrical Engineering and Computer Science students' use and perception of artificial intelligence (AI) tools in a programming course, and their preferences for future development.

Design/methodology/approach – We conducted an anonymous exploratory survey, consisting of items with predefined response options and open-ended items. Responses to the former and open-ended items were analyzed using descriptive statistics and inductive thematic analysis, respectively. Additionally, we clustered students using the Affinity Propagation algorithm based on their expressed preferences for possible improvements of AI tools

Findings – The findings show that AI tools are not universally effective, with seven student clusters identified based on differing needs and expectations. Students expressed a need for AI tools that offer more detailed error explanations and guidance rather than just delivering correct solutions. The most common concern among students is the provision of correct solutions without adequate explanations of the underlying mistakes, leading to a lack of deeper understanding

Originality/value – This study takes an exploratory approach by examining students' perceptions and preferences for the design and capabilities of AI tools in helping them learn programming. Clustering students by preferences reveals distinct approaches that may be needed for different groups of learners. Given the limited research on such desires or on applying clustering to them, our analysis offers valuable insights into distinct viewpoints that can guide the design of future personalized educational AI tools

Keywords Programming education, Novice programmers, Generative artificial intelligence, Student perceptions, Educational technology

Paper type Research article

Introduction

Since the public release of generative artificial intelligence (GenAI) tools, such as ChatGPT, in 2022, students have increasingly turned to these technologies for assistance in learning and completing coursework, especially in programming. These tools, powered by large language models (LLMs), have demonstrated their potential to provide immediate responses to coding queries, generate code, and explain errors, revolutionizing how students engage with their studies. Studies by Balabdaoui *et al.* (2024), Garrel and Mayer (2023) have shown that the use of GenAI tools is prevalent in engineering and computer programming domains, underscoring the need to further investigate these students' motivation and future expectations. However, while many studies have focused on integrating these tools into programming courses (Frankford *et al.*, 2024; Kazemitabaar *et al.*, 2023; MacNeil *et al.*, 2023; Phung *et al.*, 2023; Phung *et al.*, 2024; Sheese *et al.*, 2024; Sun *et al.*, 2024; Vadaparty *et al.*, 2024) and exploring students' and educators' perceptions on AI (Arslan *et al.*, 2025; Johnston *et al.*, 2024; Prather

Artificial Intelligence in Education
Vol. 2 No. 1, 2026
pp. 33-48
Emerald Publishing Limited
3049-5474
DOI 10.1108/AIIE-08-2025-0260

et al., 2023; Shata and Hartley, 2025; Zastudil *et al.*, 2023), fewer have delved into how students envision AI could improve their learning experience. In order to adhere to the ethical design of future AI tools to enhance learning experiences, students' personal preferences and expectations should be taken into account (Stahl *et al.*, 2022).

This paper presents the results of an exploratory survey conducted with first-year Electrical Engineering and Computer Science students enrolled in an introductory programming course. While the study is exploratory in nature and does not test formal hypotheses, we anticipated that students' use of GenAI tools might vary in purpose, and that their expectations for AI support would reflect diverse learning needs and preferences. Specifically, we address two research questions:

RQ1. How and why did the students use GenAI tools during their coursework for the introductory programming course?

RQ2. What do students expect from GenAI tools to help them learn and improve their programming skills?

Through this study, we provide insights into student perceptions, propose improvements to current AI tools, and offer recommendations for integrating AI more effectively into learning environments. This work contributes to the growing body of research on the role of GenAI in education. We identify seven groups of students based on their improvement suggestions for AI as a learning aid. This highlights the diverse learning needs students have regarding GenAI, emphasizing the importance of aligning AI tool development with student needs and ethical considerations. This preliminary study informs the future design of personalized, adaptive AI tools for learning that align with students' expectations.

Related work

A wide range of studies (Kazemitabaar *et al.*, 2023; MacNeil *et al.*, 2023; Sosnovsky *et al.*, 2025; Van Petegem *et al.*, 2023) has examined programming learning aids, focusing on how students learn programming and how tools can support this process. We summarize this work by grouping it into several categories discussed below.

Programming learning aids

A variety of learning aids have been developed to support students in learning programming. These tools aim to provide guidance, feedback, and personalized learning experiences through different forms of interaction. One group of systems (Alpizar-Chacon *et al.*, 2020; Brusilovsky *et al.*, 2018; Hoq *et al.*, 2025; Schez-Sobrino *et al.*, 2020; Van Petegem *et al.*, 2023) focuses on offering learning content designed by tutors, which can be personalized through recommendations or manually adapted. Such learning content often includes worked examples, practice tasks, or visual and interactive materials, such as code visualization or interactive textbooks. These approaches support self-paced learning and adapting to students' performance patterns. This research area is being transformed by GenAI, which opens new possibilities for making learning content even more engaging (Sosnovsky *et al.*, 2025). Another category of learning aids includes systems that analyze students' code, ranging from tools for automated assessment to more advanced approaches for code analysis and strategy inference. Automated assessment systems typically evaluate program correctness and provide immediate feedback on task performance (Das *et al.*, 2016; Van Petegem *et al.*, 2023). These tools are widely adopted in programming classes and can handle large numbers of submissions. However, the feedback is typically limited to technical correctness and may fail to address students' underlying misconceptions. Several research papers (Leite and Blanco, 2020; Paaβen *et al.*, 2016) have explored strategy inference and error identification to better understand students' problem-solving processes. These approaches are focused on research or experimental settings, but they provide valuable insights that could support broader

educational adoption in the future. Some systems also integrate question-and-answer modules that allow students to post questions and receive answers from tutors (Gao *et al.*, 2022; Van Petegem *et al.*, 2023). While these systems foster learning and clarify specific doubts, they often suffer from delayed responses and varying answer quality, depending on the availability and expertise of the tutors. Despite their strengths, existing systems often leave students waiting for feedback or seeking further clarification beyond what automated systems can provide. This gap highlights the potential of GenAI tools to deliver contextualized, conversational, and adaptive feedback. Unlike traditional systems, GenAI can respond immediately to students' specific questions, explain errors in natural language, and adapt explanations to individual learning needs. This has made them central in recent educational practice, which also motivates our study.

GenAI as learning aid

GenAI tools have shown promising results in helping novice programmers in various ways. They can explain code (Lekshmi-Narayanan *et al.*, 2024; MacNeil *et al.*, 2023), correct students' code, and provide detailed explanations of the made changes (Phung *et al.*, 2023), or only one-sentence hints indicating what the student should pay attention to correct the mistake on their own (Phung *et al.*, 2024). They can also generate complete programming code (Kazemitabaar *et al.*, 2023) or serve as an AI assistant, answering students' questions and providing help with coding (Sheese *et al.*, 2024; Sun *et al.*, 2024), and generating feedback on their implemented code (Frankford *et al.*, 2024). Formulating the confusion they might be facing into a meaningful prompt can be helpful. Given the enormous potential of GenAI to support novice programmers, it is essential to closely monitor students' use and requirements for such tools to learn programming.

Some studies have investigated how students interact with GenAI tools. Sheese *et al.* (2024) have analyzed the types of interaction with GenAI integrated within a programming environment. The most frequent type of interaction was seeking help with debugging their code, followed by assistance with code implementation. Fenu *et al.* (2024) have identified five types of students based on their interaction with an AI system, with more than half being the Passive problem-solving type and others being Error/Optimization-driven, Guide-driven, Independent, or Interactive problem-solving type. In the study by Kazemitabaar *et al.* (2023), almost half of the tasks students used GenAI for were copied from GenAI without modification. Students have reported in surveys that they use GenAI tools to debug and generate code (Balabdaoui *et al.*, 2024; Prather *et al.*, 2023) or to create explanations of code and concepts (Zastudil *et al.*, 2023). These findings indicate that not all students use GenAI tools for the same purposes and that their expectations might differ.

Perceived areas for improvement in GenAI tools

Limited research has investigated students' perceptions regarding potential improvements in current GenAI tools or their general expectations for these tools to assist with programming. Some studies, for example, by Balabdaoui *et al.* (2024), consider students' expectations as their thoughts on future adoption and the impact of GenAI tools on education. The expectations we refer to in this paper relate to students' personal preferences for its design and capabilities in helping them learn programming. Rienties *et al.* (2025) have explored the perception of distance learning students of a hypothetical AI Digital Assistant. Students would like the assistant to provide 24/7 support for various academic tasks and to adapt to each student, offering insights into their knowledge gaps and immediate feedback on their learning activities. Some of them would also find emotional and social support helpful. Students have provided valuable feedback on AI-powered tutors, highlighting specific areas for improvement. In the study by Frankford *et al.* (2024), for example, students expressed the need to be pointed directly to the areas of code that need improvement, to prevent over-reliance during learning, and to provide more examples. Similarly, a study by Denny *et al.* (2024) has

revealed students' desire to arrive at solutions independently rather than be given answers directly, and to receive responses aligned with their level of knowledge, indicating the need for AI tools to understand student models to provide personalized guidance.

Relation between GenAI usage and learning outcomes

Despite their vast potential to enhance students' programming learning, the long-term impact of various types of GenAI use on the overall learning process remains unclear. Research on the short-term effects of GenAI use yields mixed findings. Some of them agree that using GenAI tools can improve students' performance (Kazemitabaar *et al.*, 2023; Prather *et al.*, 2024; Sun *et al.*, 2024). A study by Yilmaz and Yilmaz (2023) has found positive effects of ChatGPT use on students' computational thinking skills, programming self-efficacy, and motivation. It is important to note that students were required to formulate the prompts independently based on the programming task presented as a Unified Modeling Language (UML) diagram, since ChatGPT was unable to process images. On the other hand, Jošt *et al.* (2024) found that average reliance on GenAI and its use for tasks demanding critical thinking, including debugging, are related to lower final grades. The search for additional explanations from GenAI did not significantly relate to the final performance. Prather *et al.* (2024) found it beneficial for some students but detrimental for others. These conflicting findings may be influenced by the programming languages used in the studies, students' prior knowledge, or other factors related to individual differences and approaches to using these tools. However, it implies that educational AI tools must be carefully designed to ensure safe and effective integration for all students. By examining students' motivations and patterns of use, we can more effectively assess the effects, both positive and negative, of various AI tools in education.

Method

We conducted an online survey via Google Forms, collecting responses from May 28th to June 13th, 2024. The survey link was distributed to first-year students who had been enrolled in the Introduction to Programming course at the Faculty of Electrical Engineering and Computer Science at the University of Sarajevo in the previous semester, where foundational programming concepts are taught in the C language. The study was conducted in Sarajevo, Bosnia and Herzegovina. Participation was voluntary and anonymous, encouraging honest feedback, and students could complete the survey at their convenience. A total of 61 students participated, comprising 24 females and 37 males, aged 18–22 years. Of the participants, 53 were in their first year, while eight had previously enrolled. At the outset, participants were informed that their anonymized responses would be used in a research project to improve the learning process. The questionnaire items were designed to ensure that no personally identifiable information is collected. The complete workflow of the study, covering all methodological steps, is illustrated in Figure 1.

Questionnaire

The survey consisted of 9 questions to elicit clear responses to our research questions; 2 were open-ended. We were interested in understanding how often students used ChatGPT or similar tools for help in the course, what kind of help they requested, and in which situations. Additionally, we sought to determine which type of AI tool they thought would help them learn programming more effectively or efficiently. For two questions (type and situations of use), we offered possible choices inspired by students' responses to similar questionnaires in related work (Prather *et al.*, 2023; Zastudil *et al.*, 2023), with an additional optional field for students to write a new, different response. Students could have chosen all the response options that applied to their case. To better understand students' learning struggles, we asked them an open question about which programming concept was the hardest for them to learn. The complete list of survey questions is given in the Appendix.

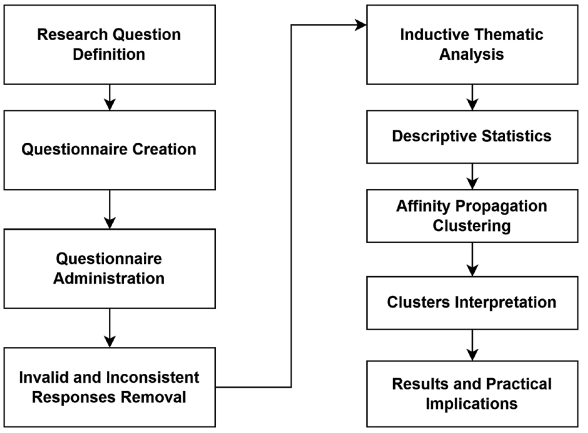

Figure 1. Summary of the study procedures, from survey creation to clustering and analysis of responses. Source: Authors' own work

Data analysis

The results were first analyzed to remove invalid responses. For instance, the question related to the type of use had an option I did not use ChatGPT or similar tools as help in the course, which cannot be selected together with other options (a participant either used it for something or did not use it at all). After removing all responses from participants who demonstrated such inconsistencies, the final number of 56 valid submissions was determined.

All new responses to the two questions with possible choices were carefully reviewed to determine whether they aligned with existing possible choices or introduced new ones. The answers to the question about the type of use did not introduce any new answer categories, while a new category When working on tasks for which I did not know C syntax well was identified in one student's answer to the question about situations they used AI tools in. For two open-ended questions, we employed inductive thematic analysis, an established methodology (Braun and Clarke, 2006), to identify meaningful categories and assign responses accordingly. Each response could be categorized into multiple groups based on its content. The categories identified for both questions are shown in Table 1. Given that this is an exploratory survey, we

Table 1. Categorization of open-ended question responses

Hardest concepts in C	Possible improvements of AI
Arrays	Possible, but no proposition
Operators	AI cannot help
Structures	Explaining compiler errors
Unknown	AI tool only for programming
Syntax	Explaining how code works in detail
Problem solving logic	Answering questions
Nothing	Giving an idea of how to approach a problem
Code printout	Not sure
Matrices	Empty response
Strings	Not giving concrete code
Dynamic allocation	Finding/explaining what is wrong in code
File handling	Nothing more beyond what current tools can do
Pointers	

Source(s): Authors' own work

qualitatively analyze the answers to all the questions, reporting percentage distributions for all of them, and descriptive statistics for the quantitative questions related to frequency of use. The analysis was performed using Python, which served as the main tool for data cleaning and visualization. The plot-likert library was used for visualizing Likert-scale items, and scipy and scikit-learn for analysis and clustering. Additionally, we run a clustering algorithm on students' responses to the question about future improvements of AI tools to identify differences in their needs. The clustering procedure is described in detail in the following section.

Clustering procedure

Students expressed their preferences for possible improvements to AI tools to better serve as a learning aid. The categories from Table 1, which were identified from their responses, were then coded with numbers from 0 to 11. Each student's response was represented as a list of these numbers. We aimed to identify a subset of representative categories (students' preferences) from the answers, with the exact number of clusters unknown in advance. Therefore, we applied the Affinity Propagation clustering algorithm (Frey and Dueck, 2007) with the Jaccard similarity measure between the students' responses. The algorithm implementation from the Python scikit-learn library was used, with a damping parameter equal to 0.8 and 500 iterations. The values {0.5, 0.7, 0.8, 0.95} for the damping parameter and {200, 500, 1,000} for the number of iterations were tested, but other values did not yield meaningful categorization, or the categorization was essentially the same as with the chosen parameters.

Limitations

As with any survey-based research, this study's findings are subject to certain limitations. The data were collected within a relatively short time frame (17 days), which may have constrained the depth of responses and introduced temporal bias. The self-reported data may not fully capture students' actual behavior, as participants might not accurately recall or may underreport specific uses of AI tools, especially given the sensitivity around academic integrity. Additionally, the sample is limited to first-year electrical engineering and computer science students from a single university, which could restrict the generalizability of the findings to other disciplines or regions. Future studies will explore more diverse student populations and examine longitudinal impacts to better understand how reliance on GenAI tools evolves.

It is important to note that the questionnaire used in this study was not subjected to formal validation procedures. This limitation stems from the study's exploratory design, which was primarily concerned with generating initial insights into students' visions for improving AI tools. In exploratory studies, the emphasis is on identifying potential relationships and formulating hypotheses, rather than confirming them with highly validated instruments (Maxwell, 2013). As such, the questionnaire served as a tool for initial data gathering, providing a foundation for future research that may incorporate more rigorous measurement and validation techniques.

Results

Quantitative insights

Frequency of use. All participants had heard about the ChatGPT tool. Almost all students (95%) reported using ChatGPT or similar tools to some extent for help in the course. Most students (66%) reported a frequency of use of 3 or higher. The most frequently reported value (Mode) was 3, with an interquartile range (IQR) of 2–4. The percentage distribution for all possible answer categories is shown in Figure 2. A significantly higher number of students took the course for the first time ($n = 49$) than those who had taken it before ($n = 7$). Regarding

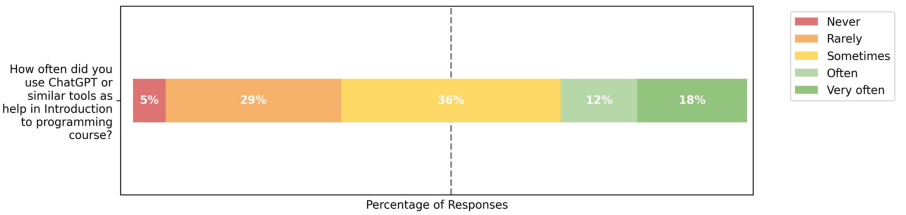

Figure 2. Response distribution regarding the frequency of ChatGPT use. Source: Authors' own work

the difference in use, all seven students who had already taken the course used ChatGPT or similar tools for help.

Qualitative insights

Type of Use. Students mainly used AI tools to debug their code (82%), explain compiler errors (68%), and clarify unclear programming concepts (57%). Figure 3 shows the distribution of the response percentages. These findings align with previous studies, which found that debugging is the most common reason for seeking AI assistance. Notably, over half of the students used AI tools to ask questions about programming concepts, a higher-than-expected result compared to other use cases. Additionally, some students used AI tools to generate suggestions for starting coding projects or to produce complete program solutions for assignments.

Reason of Use. Students usually used AI tools for help when they worked on tasks that were not clear to them (71%), when they could not ask someone else for help (46%), or when they did not have enough time to do the task on their own (32%). Contrary to the research conducted by Zastudil *et al.* (2023), finishing busy work with AI tools was ranked much lower in our survey. Only 22% of the students stated they used them when working on tasks they did not find important or that did not bring them enough credits compared to the required effort. A new reason a student brought up for using AI tools is working on tasks for which they do not know C syntax well. Figure 4 shows the distribution of the answers regarding all the mentioned situations.

Learning Struggles. The most difficult concepts for students to learn in the course were *pointers* (45%), followed by file handling (16%), and dynamic allocation and strings (13% each). The percentage distributions are shown in Figure 5. We did not find any clear relationship between the perceived most complex concepts and the other student responses. However, we note that students who had already taken the course reported struggling with four

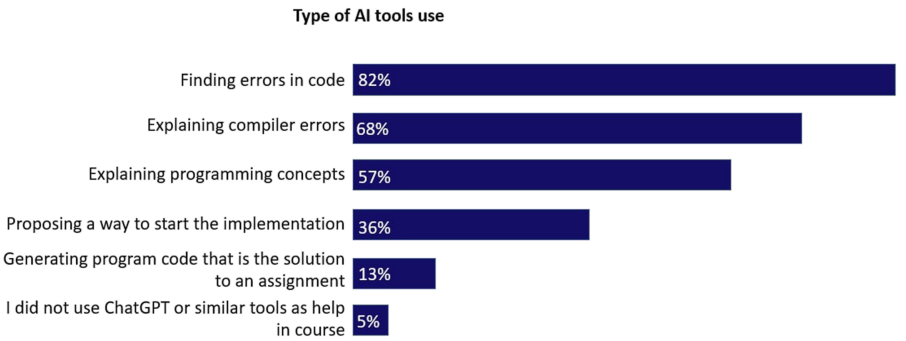

Figure 3. Students' responses regarding the type of AI tools use in the course. Source: Authors' own work

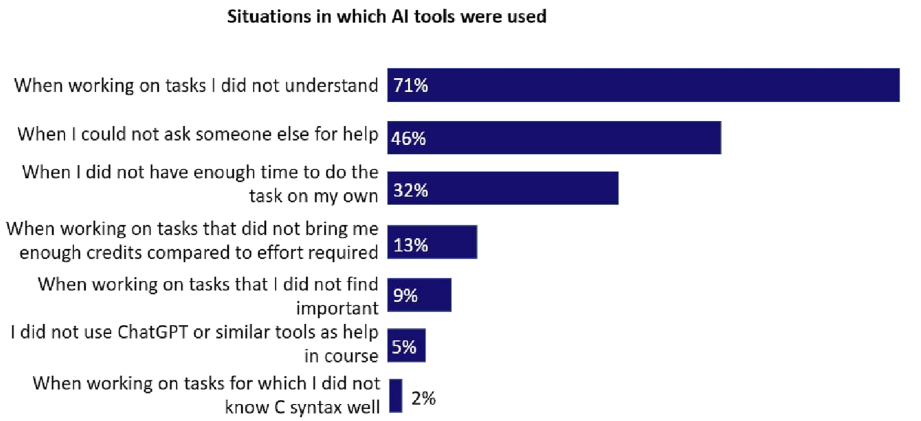

Figure 4. Students' responses regarding the situations in which they used AI tools. Source: Authors' own work

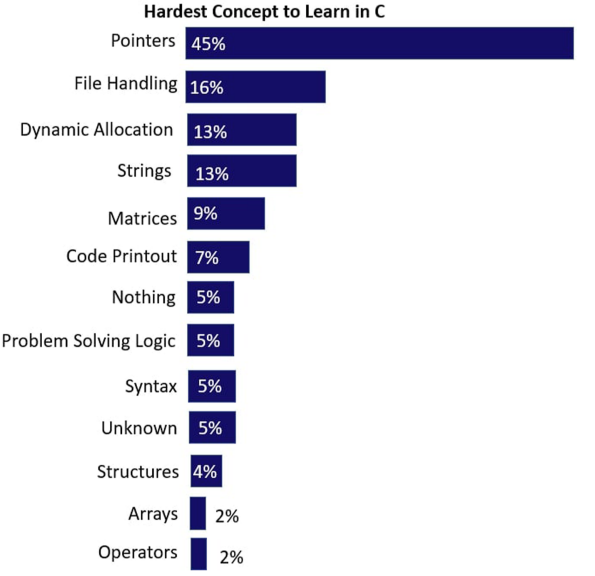

Figure 5. Students' responses regarding the hardest concept they had to learn in the C programming language. Source: Authors' own work

concepts, mostly *pointers* and strings, followed by *code printouts* and *operators*. These results can be valuable to educators, who can potentially adapt the learning materials and consider the students' struggles.

AI Tools as Help in Learning. Students' opinions regarding the potential improvement of AI tools as a help in learning to program were divided. The percentage distribution among the identified categories of answers is shown in Figure 6. The Affinity Propagation clustering algorithm identified 9 student clusters, each characterized by the most representative answers. We combine the three clusters of students who answered that they were not sure, did not express any preference, or did not answer at all, into one cluster of *Indifferent students*. We

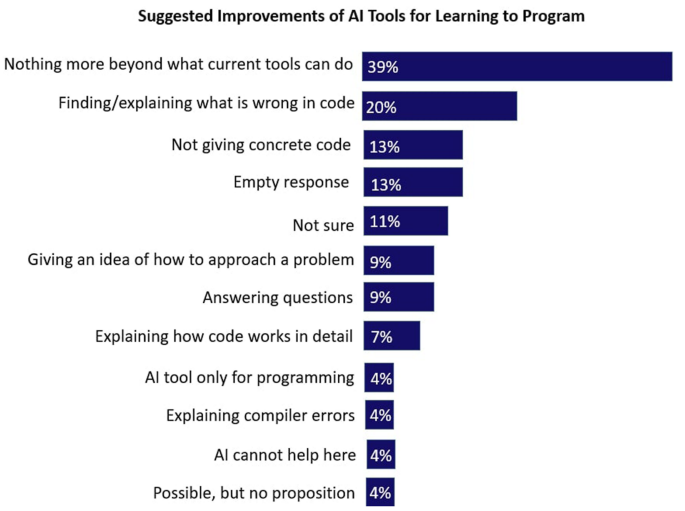

Figure 6. Students' responses regarding possible improvements of AI tools for help in learning to program. Source: Authors' own work

hereby present the final seven groups of students based on clusters of their expressed personal preferences for improving AI tools for learning: (1) Traditional Programmers, (2) AI Guide Seekers, (3) AI Insight Seekers, (4) AI Starting Point Seekers, (5) AI Debug Seekers, (6) Indifferent Users, and (7) Satisfied AI Users.

Traditional Programmers. Some students ($n = 2$) do not believe AI can help in learning to program and report that they never or rarely used AI as help in the course. These students reported that problem-solving logic, dynamic allocation, syntax, or nothing in particular was the hardest for them to learn. Considering that GenAI tools' help with problem-solving logic is later reported as non-notable, and that syntax is something that is learned over time and with practice, it is understandable that this group of students might not need additional help and can overcome the difficulties they face on their own.

AI Guide Seekers. AI Guide Seekers ($n = 3$) wish for an AI tool capable of explaining what is wrong in their programming code instead of generating the correct solution. Although some GenAI tools already have this capability, given their indeterministic nature, students cannot rely on them to consistently provide the necessary personalized guidance, rather than just revealing the correct answers. Not restricted to this group of students, 13% of the students highlighted that GenAI tools that help them learn programming should not generate complete code solutions, as they currently do. *AI Guide Seekers* students can most likely formulate a starter code for solving a problem, but need additional help in solving the errors they encounter, in such a way that they understand the mistake they made and can correct it on their own.

(1) "Potentially better explanation of the error in code and its cause, as well as compiler error interpretation, instead of offering lines of code, we could replace (i.e. when it would be more in a corrective function and not in serving the final code)."

(2) "(. . .) The only thing they do not do well is explain the problem without giving code, in the sense that they cannot give a nice pseudocode, or a template for working on tasks. (. . .)"

(3) "(. . .) but the tool should not do everything on its own, but only give guidelines, point out where our mistakes are, and similar (. . .)"

AI Insight Seekers. This group of students ($n = 3$) would find an always-available tool for answering their questions about programming or specific programming code helpful. Considering the often large number of students enrolled in programming courses, it is sometimes not feasible for professors and teaching assistants to answer all the questions each student may have. Those questions might be crucial to their understanding of some topics, so a tool that provides course-specific answers at any time would certainly be valuable.

(1) "I am sure something could be done to serve students as a teaching assistant. That tool would be able to help answer all the questions that are usually asked of the professor or teaching assistants. It does not generate the whole code or task solutions, nor does it even have that possibility."

(2) "that it is there where I need an explanation of the code or some statement, and the professor has already explained it/I did not understand the professor/I need a simpler and quicker explanation"

AI Starting Point Seekers. These students ($n = 5$) express the wish to solve the programming tasks independently, but require additional help to start implementing the solution. They state that they do not have a clear idea of how to approach a given problem or to formulate the algorithm for its correct implementation. The affinity propagation algorithm identified a common combination between this answer and the answer on AI not generating concrete code, indicating that students would like to make an effort to implement the solution on their own.

(1) "It would help me to have some basic guidelines for solving a task, because as a novice programmer, I find it significantly harder to actually start and design the concept of something, than to implement it. Lack of knowledge was an obstacle because I was usually not aware of all the available options, so I lacked ideas."

(2) "It would be nice if AI could write templates/pseudocodes to explain better how some problem is solved. (. . .) but not to do everything on its own, but rather to give guidance and hints (. . .)"

AI Debug Seekers. Some students ($n = 7$) seek debugging support from AI. They would like AI tools to identify mistakes in their programming code. These students do not emphasize that they need guidance as part of the *AI Guide Seekers* group, but rather immediate solutions to the faults they encounter in their code. From their answers, it appears they typically seek this type of help when they encounter quick-to-fix but hard-to-spot mistakes in their code.

(1) "for help in finding the errors."

(2) "Some tool that would reveal 'stupid' mistakes(. . .)"

Indifferent Users. Some students were not sure or did not provide any comments on how these tools can offer them better help in learning to program. Specifically, 4% of them expressed a desire for improvement but did not provide clear reasons or suggestions. 11% of the students stated they were unsure whether the tools could be enhanced for learning. 13% of the students did not answer this question. The students' preferences in this group ($n = 15$) should be further analyzed.

Satisfied AI Users. This group of students ($n = 21$) is prevalent among the participants. 39% of the students believe that the current capabilities of ChatGPT and similar tools they use (e.g. Copilot, Gemini, Replit) are enough to help them learn programming. However, they still raise concerns about their reliability. We note that some students express dissatisfaction with ChatGPT, but use alternative AI tools that overcome its limitations. Students in this group generally seem open to exploring different GenAI tools that can provide the type of help they need.

(1) "I think that tools like ChatGPT already explain some concepts well and can be a good help while learning to program (they quickly come up with examples for every concept). The only problem is the reliability and accuracy of all information."

(2) "Many AI chatbots are much better than ChatGPT. For example, ChatSonic could do almost any task without errors, while ChatGPT serves only as help, but cannot do anything on its own."

(3) "All tools I need are already implemented, e.g. Replit explains the code I don't understand because it has its AI engine."

Discussion

In this exploratory survey, we aimed to illuminate students' use of GenAI tools in an introductory programming course and their perspective on AI's potential as a learning aid. We found that students primarily used GenAI tools to support their programming course, with 95% rating their frequency of use between 2 and 5 on a 5-point Likert scale. Students' answers were diverse, but the most frequent use of GenAI tools was for debugging, and the most common reason for using them was not understanding a given task. This underscores the need to rethink the conclusions drawn from students' digital logs of their programming activities. Their patterns of behavior and errors might not reflect their understanding and knowledge as previously assumed.

We identified seven groups of students based on their preferences for future development of AI tools for learning. This divide in opinions related to AI tools, even among a relatively small group of students, and also noticeable in Johnston *et al.* (2024), Rienties *et al.* (2025), suggests that generic AI support may be ineffective and that students' needs should be first identified before providing them with an AI tool for learning to mitigate the negative impact on their learning process. Practical implementation should involve adaptive tools that offer variable levels of guidance depending on learners' current knowledge and preferences. Despite the diversity of student behaviors and preferences, GenAI tools have the potential to enhance programming learning across all groups. For students who struggle with conceptual understanding, GenAI can provide different levels of support, such as offering a starting point or a conceptual overview of a task, guiding them step-by-step through the problem-solving process, or intervening only when they face obstacles that are too difficult for them to solve independently. For students who already understand basic concepts, it can serve as a tool for refining solutions, exploring alternative solutions, or testing their understanding in new contexts. The identified groups of students' preferences for AI tools may be related to their current learning states, meaning they might change over time as they acquire more knowledge. We leave this exploration for future work.

Based on survey insights, we present suggestions for adopting AI tools in learning and research areas that have yet to be well investigated.

Developing GenAI tools for learning. General-purpose GenAI tools already have the abilities that some students specified as potential improvements to help them learn to program. However, they require careful prompting and sometimes still exhibit deviation from the curated prompt (Frankford *et al.*, 2024). These tools are not primarily designed for learning and risk undermining learning gains by instantly providing solutions to students' questions (Abdelghani *et al.*, 2023). Even with proper education on prompt engineering, the indeterministic nature and frequent changes to the models offered to the public always leave room for fluctuations. One way to overcome this problem is to fine-tune open-source LLMs for programming learning. These models can provide various types of assistance to students, tailoring their support to the individual learning preferences. The type of support offered could be further aligned with the student groups we identified based on their expectations from GenAI tools. For instance, they could be fine-tuned to generate answers without revealing code solutions, guiding students toward the correct solution in natural language, or only giving an idea of how to approach the problem. The likelihood of deviating from the prompt would then be highly reduced. The learning tool could also assess learning gains by generating follow-up tasks that require the student to apply the recently acquired

knowledge in a new setting. Ideally, the models could adapt dynamically, identifying the most effective type of help based on students' behavior and progress to maximize their learning gains.

Explaining AI Tool Outputs to Enhance Learning. Current GenAI tools often provide direct solutions rather than fostering a deeper understanding through explanation. This risks promoting passive learning, where students accept answers without engaging critically. Incorporating explainable AI (XAI) can shift this dynamic by generating correct code and explaining why errors occur and how to resolve them. This approach encourages students to develop problem-solving skills independently, fostering cognitive engagement. XAI tools could guide students through debugging with personalized hints, ensuring AI supports learning rather than replacing critical thinking, ultimately improving long-term understanding.

Encouraging cognitive engagement in tasks. Even with the perfect GenAI learning tool, some students would still turn to tools that provide instant solutions to their tasks if their current goal was not understood. The ability to recognize programming mistakes comes with time and experience, so it is understandable that novice programmers most often use GenAI tools in these cases. However, more harm than good can be done if these tools enter a hallucinatory state when providing an answer. Suppose students do not critically evaluate or double-check the answers generated by GenAI tools. In that case, they risk learning something incorrectly or spending even more time on the initial assignment. Therefore, it is essential to appropriately adjust the tasks given to students to encourage cognitive engagement and prevent the risk of blindly pasting solutions from GenAI tools. Critical thinking could be practiced by encouraging students to use GenAI tools to help with their tasks, then to reflect on the accuracy of their answers before providing their final answer, as done by Ding *et al.* (2023). This encourages students to critically evaluate the generated answer rather than immediately accepting it as accurate, and might help them realize that these tools are prone to error.

Non-English prompts performance research. Another possible reason for not getting the desired output from GenAI tools, even though they have the required capabilities, is the language used to prompt them. Most of the training data for GenAI tools is in English, which has been shown to lower their performance on prompts in other languages, for instance, in the NLP domain (Lai *et al.*, 2023). Students with limited English proficiency might receive less helpful responses when asking for help in their native language. However, future research is needed to confirm this hypothesis in the programming learning domain, as proposed in Kazemitabaar *et al.* (2023). Custom-developed GenAI learning tools can also be fine-tuned to support all students equally, regardless of their language proficiency. The performance of such tools should be sifted through before employing them in practice.

GenAI impact research. While some studies (Jošt *et al.*, 2024; Prather *et al.*, 2024; Xue *et al.*, 2024) have explored the role of GenAI in programming education, there is a need to investigate the distinct short - and long - term effects of two types of GenAI tools: (1) those that generate code and (2) those that guide students without providing direct solutions. Code-generating tools may boost short-term efficiency but risk promoting over-reliance and reducing cognitive engagement. In contrast, guidance tools encourage critical thinking and deeper learning by helping students troubleshoot and understand concepts. Future research should examine how these tools affect programming self-efficacy and independent problem-solving skills over time and assess their impact across diverse student populations. Longitudinal studies can reveal how different tools influence immediate learning outcomes and the development of computational thinking, an essential skill for long-term success in programming.

Conclusion

In this exploratory study, we investigated how first-year Electrical Engineering and Computer Science students use AI tools such as ChatGPT in an introductory programming course and

how they envision these tools enhancing their learning experiences. The findings provide significant initial information on the patterns, motivations, and expectations of students using AI-driven learning tools. While students frequently use these tools for debugging and error identification, there is a clear demand for AI solutions that provide more detailed explanations rather than pre-generated solutions. However, not all students want the same kind of help from AI tools. Based on their answers, we identified seven groups of students: (1) Traditional Programmers, (2) AI Guide Seekers, (3) AI Insight Seekers, (4) AI Starting Point Seekers, (5) AI Debug Seekers, (6) Indifferent Users, and (7) Satisfied AI Users. This paper presents a preliminary step toward understanding how generative AI tools can better support programming education. It informs future development in AI-assisted learning by highlighting the gap between current tool capabilities and students' learning needs. The gathered insights underscore the importance of designing AI tools that encourage cognitive engagement rather than over-reliance on code generation, thus promoting deeper learning and understanding.

Future research will investigate the long-term effects of AI tool use on programming education, focusing on how these tools impact learning gains, knowledge retention, and student engagement. Importantly, this will examine how dynamically it adapts to factors such as student personality traits and learning styles in shaping the effectiveness of AI-assisted learning. Building on the insights from this study, future work will also prioritize the development of AI-driven personalized learning systems tailored to support diverse learner profiles. These systems will aim to adapt dynamically to the students' specific needs, offering tailored guidance and feedback to enhance both the learning process and overall skill retention. By aligning the AI tool development with student needs and ethical principles, educators and developers can ensure these technologies contribute meaningfully to learning experiences while avoiding potential pitfalls.

Appendix
Survey questions

(1) Did you take the Introduction to programming course for the first time in school year 2023/2024?
- Yes
- No

(2) Please select your gender.
- Male
- Female
- I do not want to disclose.

(3) What is your age?

(4) Have you heard about the tool ChatGPT?
- Yes
- No

(5) How often did you use ChatGPT or similar tools as help in Introduction to programming course?

Never 1 2 3 4 5 **Very Often**

(6) What kind of help did you use ChatGPT or similar tools for?
- Generating program code that is the solution to an assignment.
- Explaining compiler errors.
- Finding errors in code.

- Proposing a way to start the implementation.
- Explaining programming concepts.
- I did not use ChatGPT or similar tools as help in course.
- Other:

(7) What situations did you use ChatGPT or similar tools in?

- When working on tasks that I did not understand.
- When working on tasks that I do not find important.
- When working on tasks that do not bring me enough credits compared to the effort required.
- When I could not ask someone else for help.
- When I did not have enough time to do the task on my own.
- I did not use ChatGPT or similar tools as help in the course.
- Other:

(8) What was the hardest concept in C for you to grasp?

(9) Do you think that some other kind of AI tool could help you learn some programming concepts faster or easier? You are welcome to give an example of how that tool might help you and what possibilities you would like it to have.

References

Abdelghani, R., Sauzéon, H. and Oudeyer, P.-Y. (2023), "Generative AI in the classroom: can students remain active learners?", *NeurIPS 2023 - GAIED Workshop - Conference on Neural Information Processing Systems*.

Alpizar-Chacon, I., van der Hart, M., Wiersma, Z.S., Theunissen, L.S.J. and Sosnovsky, S. (2020), "Transformation of PDF textbooks into intelligent educational resources", *CEUR Workshop Proceedings*, Vol. 2674, pp. 4-16.

Arslan, N., Haj Youssef, M. and Ghandour, R. (2025), "AI and learning experiences of international students studying in the UK: an exploratory case study", *Artificial Intelligence in Education*, Vol. 1 No. 1, pp. 1-23, doi: 10.1108/aiie-10-2024-0019.

Balabdaoui, F., Dittmann-Domenichini, N., Grosse, H., Schlienger, C. and Kortemeyer, G. (2024), "A survey on students' use of AI at a technical university", *Discover Education*, Vol. 3 No. 1, p. 51, doi: 10.1007/s44217-024-00136-4.

Braun, V. and Clarke, V. (2006), "Using thematic analysis in psychology", *Qualitative Research in Psychology*, Vol. 3 No. 2, pp. 77-101, doi: 10.1191/1478088706qp063oa.

Brusilovsky, P., Malmi, L., Hosseini, R., Guerra, J., Sirkiä, T. and Pollari-Malmi, K. (2018), "An integrated practice system for learning programming in python: design and evaluation", *Research and Practice in Technology Enhanced Learning*, Vol. 13 No. 1, p. 18, doi: 10.1186/s41039-018-0085-9.

Das, R., Ahmed, U.Z., Karkare, A. and Gulwani, S. (2016), "Prutor: a system for tutoring CS1 and collecting student programs for analysis", ArXiv abs/1608.03828.

Denny, P., MacNeil, S., Savelka, J., Porter, L. and Luxton-Reilly, A. (2024), "Desirable characteristics for AI teaching assistants in programming education", *Proceedings of the 2024 on Innovation and Technology in Computer Science Education*, Vol. 1, ACM, pp. 408-414, doi: 10.1145/3649217.3653574.

Ding, L., Li, T., Jiang, S. and Gapud, A. (2023), "Students' perceptions of using ChatGPT in a physics class as a virtual tutor", *International Journal of Educational Technology in Higher Education*, Vol. 20 No. 1, p. 63, doi: 10.1186/s41239-023-00434-1.

Fenu, G., Galici, R., Marras, M. and Reforgiato, D. (2024), "Exploring student interactions with AI in programming training", *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*, Association for Computing Machinery, Cagliari, pp. 555-560.

Frankford, E., Sauerwein, C., Bassner, P., Krusche, S. and Breu, R. (2024), "AI-tutoring in software engineering education", *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, ACM, Vol. 2, pp. 309-319, doi: 10.1145/3639474.3640061.

Frey, B.J. and Dueck, D. (2007), "Clustering by passing messages between data points", *Science*, Vol. 315 No. 5814, pp. 972-976, doi: 10.1126/science.1136800.

Gao, Z., Erickson, B., Xu, Y., Lynch, C., Heckman, S. and Barnes, T. (2022), "You asked, now what? Modeling students' help-seeking and coding actions from request to resolution", *Journal of Educational Data Mining*, Vol. 14 No. 3, pp. 109-131.

Garrel, J.V. and Mayer, J. (2023), "Artificial intelligence in studies–use of ChatGPT and AI-based tools among students in Germany", *Humanities and Social Sciences Communications*, Vol. 10 No. 1, p. 799, doi: 10.1057/s41599-023-02304-7.

Hoq, M., Patil, A., Akhuseyinoglu, K., Brusilovsky, P. and Akram, B. (2025), "An automated approach to recommending relevant worked examples for programming problems", *Proceedings of the 56th ACM Technical Symposium on Computer Science Education*, Vol. 1, pp. 527-533, ISBN 9798400705311.

Johnston, H., Wells, R.F., Shanks, E.M., Boey, T. and Parsons, B.N. (2024), "Student perspectives on the use of generative artificial intelligence technologies in higher education", *International Journal for Educational Integrity*, Vol. 20 No. 1, p. 2, doi: 10.1007/s40979-024-00149-4.

Jošt, G., Taneski, V. and Karakatič, S. (2024), "The impact of large language models on programming education and student learning outcomes", *Applied Sciences*, Vol. 14 No. 10, 4115, doi: 10.3390/app14104115.

Kazemitabaar, M., Chow, J., Ma, C.K.T., Ericson, B.J., Weintrop, D. and Grossman, T. (2023), "Studying the effect of AI code generators on supporting novice learners in introductory programming", *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, Hamburg.

Lai, V.D., Ngo, N.T., Veyseh, A.P.B., Man, H., Dernoncourt, F., Bui, T. and Nguyen, T.H. (2023), "ChatGPT beyond English: towards a comprehensive evaluation of large language models in multilingual learning", arXiv: 2304.05613.

Leite, A. and Blanco, S.A. (2020), "Effects of human vs. automatic feedback on students' understanding of AI concepts and programming style", *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, Portland, pp. 44-50, ISBN 9781450367936.

Lekshmi-Narayanan, A.-B., Oli, P., Chapagain, J., Hassany, M., Banjade, R., Brusilovsky, P. and Rus, V. (2024), "Explaining code examples in introductory programming courses: LLM vs humans", *Proceedings of the 2024 AAAI Conference on Artificial Intelligence, Proceedings of Machine Learning Research*, Vol. 257, pp. 107-117.

MacNeil, S., Tran, A., Hellas, A., Kim, J., Sarsa, S., Denny, P., Bernstein, S. and Leinonen, J. (2023), "Experiences from using code explanations generated by large language models in a web software development e-book", *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, Association for Computing Machinery, Toronto, Vol. 1, pp. 931-937.

Maxwell, J.A. (2013), *Qualitative Research Design: an Interactive Approach. Applied Social Research Methods*, SAGE Publications, Los Angeles.

Paaβen, B., Jensen, J. and Hammer, B. (2016), "Execution traces as a powerful data representation for intelligent tutoring systems for programming", *Proceedings of the 9th International Conference on Educational Data Mining, International Educational Data Mining Society*, Raleigh, pp. 183-190.

Phung, T., Cambronero, J., Gulwani, S., Kohn, T., Majumdar, R., Singla, A. and Soares, G. (2023), "Generating high-precision feedback for programming syntax errors using large language models", *Proceedings of the 16th International Conference on Educational Data Mining*, International Educational Data Mining Society, Bengaluru, pp. 370-377.

Phung, T., Pădurean, V.-A., Singh, A., Brooks, C., Cambronero, J., Gulwani, S., Singla, A. and Soares, G. (2024), "Automating human tutor-style programming feedback: leveraging GPT-4 tutor model for hint generation and GPT-3.5 student model for hint validation", *Proceedings of the 14th Learning Analytics and Knowledge Conference*, Association for Computing Machinery, Kyoto, pp. 12-23.

Prather, J., Denny, P., Leinonen, J., Becker, B.A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B.N. and Savelka, J. (2023), "The robots are here: navigating the generative AI revolution in computing education", *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*, Association for Computing Machinery, New York, NY, pp. 108-159.

Prather, J., Reeves, B.N., Leinonen, J., MacNeil, S., Randrianasolo, A.S., Becker, B.A., Kimmel, B., Wright, J. and Briggs, B. (2024), "The widening gap: the benefits and harms of generative AI for novice programmers", *Proceedings of the 2024 ACM Conference on International Computing Education Research*, Association for Computing Machinery, Melbourne, VI, Vol. 1, pp. 469-486.

Rienties, B., Duttaroy, S., Domingue, J., Herodotou, C., Tessarolo, F. and Whitelock, D. (2025), "What distance learning students want from an AI digital assistant", *Distance Education*, Vol. 46 No. 2, pp. 173-189, doi: 10.1080/01587919.2024.2338717.

Schez-Sobrino, S., Gmez-Portes, C., Vallejo, D., Glez-Morcillo, C. and Redondo, M.Á. (2020), "An intelligent tutoring system to facilitate the learning of programming through the usage of dynamic graphic visualizations", *Applied Sciences*, Vol. 10 No. 4, p. 1518, ISSN 2076-3417, doi: 10.3390/app10041518.

Shata, A. and Hartley, K. (2025), "Artificial intelligence and communication technologies in academia: faculty perceptions and the adoption of generative AI", *International Journal of Educational Technology in Higher Education*, Vol. 22 No. 1, p. 14, doi: 10.1186/s41239-025-00511-7.

Sheese, B., Liffiton, M., Savelka, J. and Denny, P. (2024), "Patterns of student help-seeking when using a large language model-powered programming assistant", *Proceedings of the 26th Australasian Computing Education Conference*, Association for Computing Machinery, New York, NY, pp. 49-57.

Sosnovsky, S., Brusilovsky, P. and Lan, A. (2025), "Intelligent textbooks", *International Journal of Artificial Intelligence in Education*, Vol. 35 No. 3, pp. 1-20, doi: 10.1007/s40593-024-00451-9.

Stahl, B.C., Schroeder, D. and Rodrigues, R. (2022), *Ethics of Artificial Intelligence: Case Studies and Options for Addressing Ethical Challenges. SpringerBriefs in Research and Innovation Governance*, Springer Nature Switzerland AG, Cham, pp. 95-103.

Sun, D., Boudouaia, A., Zhu, C. and Li, Y. (2024), "Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study", *International Journal of Educational Technology in Higher Education*, Vol. 21 No. 1, p. 14, doi: 10.1186/s41239-024-00446-5.

Vadaparty, A., Zingaro, D., Smith, D.H. IV, Padala, M., Alvarado, C., Gorson Benario, J. and Porter, L. (2024), "CS1-LLM: integrating LLMs into CS1 instruction", *Proceedings of the 2024 on Innovation and Technology in Computer Science Education*, Vol. 1, Association for Computing Machinery, Milan, pp. 297-303.

Van Petegem, C., Maertens, R., Strijbol, N., Van Renterghem, J., Van der Jeugt, F., De Wever, B., Dawyndt, P. and Mesuere, B. (2023), "Dodona: learn to code with a virtual co-teacher that supports active learning", *SoftwareX*, Vol. 24, 101578, ISSN 2352-7110, doi: 10.1016/j.softx.2023.101578.

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

Yilmaz, R. and Yilmaz, F.G.K. (2023), "The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation", *Computers and Education: Artificial Intelligence*, Vol. 4, 100147, doi: 10.1016/j.caeai.2023.100147.

Zastudil, C., Rogalska, M., Kapp, C., Vaughn, J. and MacNeil, S. (2023), "Generative AI in computing education: perspectives of students and instructors", *2023 IEEE Frontiers in Education Conference (FIE)*, pp. 1-9.

Corresponding author
Mubina Kamberovic can be contacted at: mkamberovi1@etf.unsa.ba