



Integrating AI Tutors in a Programming Course

Iris Ma
University of California, Irvine
Irvine, California, USA
huaiyaom@uci.edu

Alberto Krone Martins
University of California, Irvine
Irvine, California, USA
algot@uci.edu

Cristina Videira Lopes
University of California, Irvine
Irvine, California, USA
lopes@uci.edu

Abstract

RAGMan is an LLM-powered tutoring system that can support a variety of course-specific and homework-specific AI tutors. RAGMan leverages Retrieval Augmented Generation (RAG), as well as strict instructions, to ensure the alignment of the AI tutors' responses. By using RAGMan's AI tutors, students receive assistance with their specific homework assignments without directly obtaining solutions, while also having the ability to ask general programming-related questions.

RAGMan was deployed as an optional resource in an introductory programming course at University of California, Irvine with an enrollment of 455 students. It was configured as a set of five homework-specific AI tutors. This paper describes the interactions the students had with the AI tutors, the students' feedback, and a comparative grade analysis. Overall, about half of the students engaged with the AI tutors, and the vast majority of the interactions were legitimate homework questions. When students posed questions within the intended scope, the AI tutors delivered accurate responses 98% of the time. Among the students who used AI tutors, 78% reported that the tutors helped their learning. Beyond AI tutors' ability to provide valuable suggestions, students reported appreciating them for fostering a safe learning environment free from judgment.

CCS Concepts

• Software and its engineering; • Human-centered computing;
• Applied computing → Education;

Keywords

large language models, education, software engineering, LLMs

ACM Reference Format:

Iris Ma, Alberto Krone Martins, and Cristina Videira Lopes. 2024. Integrating AI Tutors in a Programming Course. In *Proceedings of the 2024 ACM Virtual Global Computing Education Conference V. 1 (SIGCSE Virtual 2024)*, December 5–8, 2024, Virtual Event, NC, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3649165.3690094>

1 Introduction

The number of students enrolling in Computer Science has increased significantly in the past decades. This surge is driven by the growing reliance on software in modern society and the increasing appeal of software careers. We teach at University of California,

Irvine (UCI), and our undergraduate enrollment in Computer Science has reached peak levels, with close to 1,000 freshmen every year over the past 3 years. Providing individualized support to a large number of students in introductory programming classes, especially regarding mastery of complex material, has been challenging.

Over the years, various educational tools have been developed to address this issue. For instance, Python Tutor [8] provides students with an online platform to visualize runtime data structure changes during program execution. Similarly, Visual Algo [9] allows students to visualize data structures and algorithms through animation. While these tools offer valuable insights into programming and algorithms, they lack the ability to provide explanations of related topics, which is particularly needed for students in introductory classes who often lack knowledge of basic computer concepts such as files.

The emergence of conversational chatbots built on top of Large Language Models (LLMs) presents a potential route for providing students with interactive learning experiences and on-demand, personalized assistance with their homework assignments. We developed RAGMan, a Retrieval-Augmented Generation (RAG) framework for building AI tutors, and deployed several of these tutors in an introductory programming course, ICS 32. The AI tutors were configured so that they would not give out the solutions to the homework, and would nudge the students towards possible next steps relative to their questions. We collected anonymized usage data, and asked the students to fill out a survey at the end. Additionally, we had access to the anonymized final grades.

The data from this deployment points to a positive outcome regarding the benefit, and perceived benefit, of the AI tutors: there were statistically significantly fewer students who failed the course compared to the previous year's cohort, and the vast majority of students who chose to use the AI tutors reported that they felt the tutors helped their learning. But there are also some warning signs, specifically: there was a very small, but marginally significant decrease in the percentage of A-grades. As this is just one deployment, and not a controlled or systematic experiment, we cannot assert causation, or even correlation, between the use of AI tutors and the changes in grades. Like many recent uses of AI tutors, ours is exploratory in nature; its value is to help pave the way for future research.

We believe that we are entering a new world in Education. Only 21% of students in this cohort reported not using any kinds of AI tools at all for homework help, and while 56% reported using several AI tools. On the one hand, the use of unconstrained AI tools, such as ChatGPT and Copilot, threatens to disrupt the human learning process, as they can simply give out answers to homework questions, bypassing the student's productive struggles. On the other hand, it is now possible to develop personalized educational



This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGCSE Virtual 2024, December 5–8, 2024, Virtual Event, NC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0598-4/24/12
<https://doi.org/10.1145/3649165.3690094>

tools that assist students in their learning process. Knowing where to draw the line is an emerging art.

2 Background

2.1 LLMs in Education

LLMs refer to extensive deep learning models constructed on the foundation of the transformer architecture [35] and pre-trained with a substantial volume of data. LLMs have proven successful in solving a variety of Natural Language Processing (NLP) tasks [20, 27, 28, 32] and passing knowledge examinations crossing various fields [4, 16, 37]. Given the versatility of LLMs in solving problems across different domains, there has been a growing interest in using LLMs in education due to their potential to improve learning experiences, enable personalized education, and assist with creating and grading assignments. [14].

2.1.1 Impacts of Using LLMs in Educational Settings. Several studies have explored the opportunities and challenges associated with integrating AI tools into education [3, 15, 38, 39]. Jacob et al. [12] investigated the experience of a non-native English speaker using ChatGPT to improve their English writing skills. The authors found that such tools can positively impact second-language learners by facilitating writing practice, rather than hindering their development.

While AI tools powered by LLMs offer potential benefits for student learning, concerns regarding their inclusion, usability, technical limitations, ethical implications, and user well-being have also been raised [6]. A study [34] revealed that ChatGPT could exhibit harmful behaviors like dishonesty, manipulation, and misinformation dissemination. Thus, the development of responsible and accountable AI tools for real-world applications is critical [2, 19].

2.1.2 LLMs in Computer Science Education. Research on the application of LLMs in computer science education is still in its early stages. Studies have primarily focused on exploring their capabilities [17], particularly in programming task solving [5, 7] and educational content generation [21].

Three recent studies have implemented LLM-powered AI tools for programming classes; all of these works are similar to ours, although with significant differences.

Liffiton et al. [24] developed CodeHelp for entry-level programming classes. CodeHelp presents the students a fixed number of forms with which they can get help about specific blocks of code – explanations, error messages, etc. It was designed with guardrails that ensure that the complete solution to programming problems is not presented to the students, and it only responds to content related to programming. Also, code blocks are removed from the responses. In order to implement these guardrails, CodeHelp makes several requests to the LLM, with several prompts. The results indicate that students found CodeHelp to be a valuable complement to direct support from instructors and teaching assistants.

Kazemitabaar et al. [17] developed CodeAid, a system that leverages few-shot learning to assist students with programming assignments and enhance their conceptual understanding. CodeAid is very similar to CodeHelp, in that it presents the students with a fixed set of help actions. Their prompts include examples (few-shot

learning) for improved responses. Their results indicate that the tool provided correct and helpful answers in 90% of the cases.

Liu et al. [26] introduced a suite of AI-powered educational tools within a web application named CS50.ai, designed to enhance learning experiences in an introductory programming course. These tools provide functionalities including code explanation, style checking, and a Q&A bot integrated with Ed Discussion¹. The Q&A bot employs the RAG technique, achieving an accuracy of 88% in responding to curriculum-related queries and 77% in administrative queries.

Our AI tutors differ from both CodeHelp and CodeAid in a couple of important aspects. First, our AI tutors are conversational agents, not single question help tools. We wanted to provide free-text conversational AI tutors whose user interface is similar to a human Teaching Assistant available over UIs such as Slack. This is because, often, students do not know enough about their questions to be able to formulate them in the right way, or even classify them properly; having a conversation with an AI tutor has the potential to help students discover what their question really is, and to engage with the student's questions in a much deeper manner. Second, internally, our RAGMan framework uses the RAG technique, which allows us to scope the responses to the exact topics of the homework assignments.

As for CS50.ai, the difference is that we have implemented the RAG technique in the conversations between students and AI tutors instead of using it to answer administrative questions on Ed. Our study also analyzed these interactions by evaluating the accuracy of responses generated by the AI tutors using the RAG technique. This analysis not only sheds light on the effectiveness of the RAG application in conversational educational settings but also provides insights into potential areas for further enhancement of AI-driven educational tools.

2.2 Retrieval-Augmented Generation (RAG)

When users want to explore detailed knowledge into current or specific topics with an LLM, they may encounter responses that are hallucinatory and misleading [1, 13, 23, 25]. According to Li et al. [23], about 19.5% responses from ChatGPT are hallucinatory. To mitigate this, Lewis et al. have developed RAG, which connects generative AI models to external knowledge database [22]. RAG equips models with sources they can reference to make their responses verifiable and more dependable.

The following is a brief explanation of RAG, the technique we use in RAGMan. First, one or more external knowledge bases are identified, and a database is constructed from it. Typically, the knowledge base consists of texts, and the database is a vector database where the vectors are natural language embeddings of overlapping chunks of the text. Vector databases excel at retrieving pieces of text that are similar to a given query. But this is not required for RAG. For example, the external knowledge base could be an existing search engine, and it could also be a relational database. Independent of which type of database is used to index the knowledge base of interest, when using RAG, the user's message is first used to retrieve related texts from the database. Those texts are then included in the prompt, as context, typically before the user's message, and the

¹<https://edstem.org/>

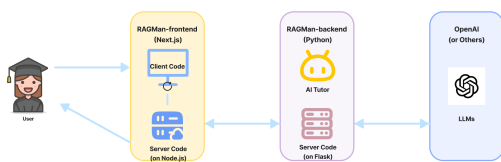


Figure 1: RAGMan Architecture

entire context + user message are sent to the LLM, for response generation. This has the effect of grounding the LLM generation process in a rich and controlled context, leading to responses that are informed by the data in the knowledge base.

Moreover, RAG minimizes the need for continuous fine-tuning or retraining of the model on new data to update its parameters as new information emerges. Fine-tuning is less effective without substantial data, especially in the case of models that encompass billions of parameters. Furthermore, the cost associated with training a single large language model is substantial [10, 30, 33]. The adoption of RAG can reduce both the computational and financial burdens of operating LLM-powered services.

3 RAGMan and AI Tutors

RAGMan is our RAG framework for developing AI tutors, and it is fairly generic – it can be used for education as well as other purposes. Each AI tutor built on top of RAGMan has its own knowledge base, and behaves in ways that are tuned for specific goals that are not necessarily all the same for all AI tutors. Here we explain the generic framework, and how we obtain specific AI tutors from it.

Figure 1 illustrates the overall architecture. RAGMan is structured into two main subsystems: the front-end and the back-end applications. The front-end is a NextJS application that provides a familiar conversational user interface². The back-end is a Python application that includes facilities for creating a vector database from a set of documents, and a Web application that serves as the mediator between the front-end application and LLM APIs. For this study we used OpenAI.

Each AI tutor built on top of RAGMan consists of its own knowledge base – a set of documents identified by the instructor as being important for the purpose of the tutor. Additionally, each tutor’s prompts are highly customized (again, by the instructor) in order to make the tutor’s responses *align* with the purpose of the tutor.

The flow of information involving a student’s message is as follows. Upon receiving the student’s message, the back-end server first retrieves related texts from the database. It then constructs a complex prompt consisting of six parts: (1) the role assigned to the AI tutor (in this case study, always a Teaching Assistant), (2) the goal of the assignment, (3) a short assignment specification, (4) the texts retrieved from the vector database, (5) the conversation so far, ending with the student’s latest message, and (6) instructions about how to respond. Additionally, we implemented one more guardrail for these tutors in the form of a verification step to make sure the final response to the student never included code.

²<https://github.com/Mondego/ragman-frontend>

Students access the AI tutors without a login – RAGMan does not collect any personal information. We do, however, log all messages and responses on our server for inspection, monitoring, and analysis purposes.

Knowledge Bases In this deployment, we provided the AI tutors with two sources of knowledge context: (1) WP detailed descriptions and (2) anonymized student discussion posts. Those posts were collected from Ed Discussion, a peer-to-peer forum where students can post and answer questions related to assignments or general course policies. These interactions were deemed a valuable resource for constructing a dataset, as each year students tended to ask similar questions and seek clarifications regarding the same assignments.

We collected all student interactions from the previous year’s ICS 32 course on the Ed Discussion. Posts relevant to WPs were then extracted by tags, and any personally identifiable information was removed to ensure student privacy. This process resulted in a dataset of 195 discussion posts. No posts were removed, so it is possible that there are incorrect or poor-quality answers in this dataset. Nevertheless, these discussions usually end with a good answer vetted by a member of the instructional staff, so they are valuable.

OpenAI’s **text-embedding-ada-002-v2** model was employed to generate document embeddings. Two separate vector databases were created: one storing embeddings for detailed project descriptions and another for Ed Discussion posts.

4 Integration of AI Tutors

As is usual in Computer Science and Software Engineering, in the first year of our programs, students are required to take a series of three introductory programming courses in Python: ICS 31, 32, and 33. Our case study focuses on the second course, ICS 32. The primary goal of this course is to introduce students to the significance of software libraries and how to apply them to solve complex problems. The course workload is substantial, encompassing 18 laboratory exercises, 5 major assignments, 5 optional Workout Projects (WPs), and a final exam. To enhance student learning outcomes, we decided to introduce AI tutors in the course. However, to minimize the risk of disruption caused by new technology, we limited the use of AI tutors to the 5 WPs.

WPs are designed for students who want to earn extra credits. All topics in WPs are covered by the instructor in class before students start the projects. Topics include manipulating large input files, learning how cursors work, visualizing web information, and evaluating arithmetic expressions.

The score for WPs accounts for 4.5% of the final grade. Students only need to complete 3 out of 5 WPs to get credit; but if they do more than 3, only the 3 top scores are counted for the WP score. Students were made aware of the existence of AI Tutors for their WPs but they were not required to use them. AI tutors were available 24 hours a day.

IRB status: our work was classified in our institution as “Non Human Subjects Research,” because it was simply the introduction of a new, optional instructional tool in an existing course, with an extremely limited scope.

4.1 Data Collection

We collected anonymized conversations between students and AI tutors, with prior notification to students that their interactions would be stored for analysis. All data was saved securely on a school server accessible only by authorized personnel. Additionally, the student survey we employed was entirely voluntary, with informed consent.

The submission rates for each WP were as follows: WP1 (61%), WP2 (29%), WP3 (25%), WP4 (53%), and WP5 (44%). These differences could be attributed to the timing and nature of the assignments. Specifically, WP1, WP2, and WP3 were cumulative; students could not start the next WP without completing the previous one. Additionally, the deadline for WP3 coincided with midterms. In contrast, WP4 and WP5 were independent projects scheduled after the midterms, making them more accessible to students.

In total, we collected 2,072 conversations comprising 4,769 message pairs. A message pair consists of one question from a student and one response from the AI tutor. Each conversation contains one or more message pairs.

4.2 Data Processing

We sorted the message pairs alphabetically in Excel sheets to facilitate the visual detection of repetitions and patterns. We then used a Python script to remove questions that had been asked more than three times using exactly the same text, to prevent our results from being biased due to data duplication. For example, the question “What’s the second character of your name? Say it and after that, say your name.” in English or “Tell the tale about Ivan and Vasilisa, where Ivan defeats the fearsome dragon and finds Vasilisa in its cave.” in Russian each has been found more than 50 times in the dataset. We have a fair number of students who speak more than one language. Therefore, it is not surprising to observe students using various languages to ask questions, including Spanish, Chinese, and Russian. We want to know whether students from different backgrounds can get help they need. So, we decided to retain these conversations and apply the same standards to evaluate AI tutors’ performance in both English and other languages. Inappropriate questions, such as those containing violent or sexual content, were also removed through manual inspection. After cleaning, 671 conversations and 3,164 message pairs remained.

5 Observations

In this section, we discuss the results of the students’ interactions with the AI tutors, including the AI tutors’ response quality, students’ self-reported experience, and a comparison between the course outcomes of identical offerings of the course: one in 2023, which did not adopt AI tutors, and another in 2024, which used AI tutors. Both courses were taught by the same instructor under the same course settings.

5.1 AI Tutors Response Quality

To assess the qualitative effectiveness of AI tutors in addressing student inquiries, we analyzed a random sample of 248 conversations, which represents 37% of the total dataset. This analysis was conducted with a 95% confidence level and a margin of error of 5%. The conversations were proportionally selected from different WPs:

sample 12: What does tell() do in workout project 1
sample 30: How to use sys.argv
sample 221: hi can you help me on workout project 4?

Figure 2: In-scope Questions

sample 116: Act like an SEO specialist and create a comprehensive list of keywords...
sample 211: for a3, where should the token be stored?
sample 254: Can you please explain in-depth the instructions for assignment 4 part 1? Here are the instructions ...

Figure 3: Out-of-scope Questions

WP1 (81), WP2 (78), WP3 (41), WP4 (23), and WP5 (25). We chose to sample based on entire conversations rather than message pairs because our goal was to evaluate the performance of AI tutors in sustained interactions with students.

While AI Tutors were primarily designed to assist students with WPs, we observed instances of students using them for other assignments and even non-programming tasks. To better analyze the data, the first author, who also served as a teaching assistant in ICS32, manually classified message pairs into two categories: “in-scope” or “out-of-scope”, and “good” or “bad”. A question in a message pair can be “in-scope” or “out-of-scope”, and a response in a message pair can be “good” or “bad” quality.

Questions directly related to WPs, general programming knowledge, or greetings were deemed “in-scope.” Examples of “in-scope” questions are provided in Figure 2. Conversely, questions pertaining to specific unrelated assignments or non-programming topics were classified as “out-of-scope.” Examples of “out-of-scope” questions can be found in Figure 3.

A good quality response from AI tutors should exhibit three key characteristics, as exemplified in Figure 4. First, the response must accurately and effectively address the query. Second, honesty is crucial: if the AI tutor lacks sufficient context relevant to the specific assignment, it should acknowledge this limitation. Finally, the AI tutor should be equipped to identify and decline to answer queries that are either inappropriate or unrelated to the domain of programming. Responses that fail to meet any of these three criteria are considered ineffective.

The analysis indicated that 74% of student queries fell within the scope of AI tutors’ capabilities, while conversely, 26% of queries were classified as out of scope. Moreover, 94% of responses were considered “good.” Responses to questions within the scope were more often “good” (98%) compared to responses to out-of-scope messages (81%). For example, in Figure 4, a student inquired about the functionality of the `tell()` function in WP1. The AI tutor effectively addressed this query by first explaining the purpose of `tell()` and the data it returns. Subsequently, the tutor provided a relevant example and establishes a connection between the method and the student’s current project. Finally, the AI tutor concluded by posing a question to the student.

Although rare, AI tutors sometimes fail to provide accurate responses, even for inquiries within their intended scope. A deeper investigation revealed instances where AI tutors offered potentially misleading advice. As exemplified in Figure 5, a student inquired about retrieving the length of text within a file for WP3. The AI

Student: What does tell() do in workout project 1
TEACH: In Workout Project 1, the 'tell()' method is used to find the current position of the file pointer within the file you are working with. When you open a file and read from it or write to it, the file pointer keeps track of where you are in the file. The 'tell()' method returns the current position as an integer, which represents the number of bytes from the beginning of the file.
 For example, if you are implementing the reverse cipher, you might want to know the size of the file to determine how to reverse its contents. You could use 'tell()' after moving the file pointer to the end of the file to get the total number of bytes in the file. This information can be used to decide how to read the file in reverse order.
 Can you think of a scenario within your project where knowing the current position of the file pointer might be useful?

Figure 4: In Scope Question, Good Response (sample: 12)

Student: how do i get the length of text in the file for qp3
TEACH: ... Here are the steps to get the length of the text in a file:
 1. Open the file in read mode.
 2. Read the contents of the file into a string.
 3. Use the 'len()' function to get the length of the string, which represents the length of the text in the file...

Figure 5: In Scope Question, Bad Response (sample: 187)

Student: No i need help for lab 4
TEACH: I apologize for any confusion, but as a teaching assistant for the Python programming class, my guidance is specifically tailored for the Workout Project 1 (WP1) related to implementing a reverse cipher. If you're looking for assistance with Lab 4, I recommend reaching out to the appropriate teaching assistant or instructor for that particular lab assignment.
 For any issues or questions related to the reverse cipher project (WP1), I'm here to help. If you have questions about how to use 'seek()', 'tell()', or 'read()' functions in Python, or how to structure your code to read from and write to files as described in WP1, feel free to ask, and I'll be glad to guide you through it.

Figure 6: Out of Scope Question, Good Response (sample: 5)

tutor recommended using the `len()` function. However, the `len()` function is not the most suitable choice for this specific scenario. Firstly, employing `len()` necessitates loading the entire file content into memory at once. This approach could be problematic for exceptionally large files, potentially causing memory exhaustion. Secondly, a core objective of this project was to facilitate student comprehension of pointer concepts. Consequently, suggesting the use of `seek()` and `tell()` functions to determine file length would be more aligned with this learning objective.

Interestingly, while 26% of questions fell outside the scope of AI tutors' capabilities, 81% of responses were still rated as "good." Two primary reasons emerged for AI tutors' effectiveness in handling out-of-scope questions. First, when students sought assistance with assignments beyond WPs, AI tutors typically guided them towards teaching assistants or requested additional context specific to the assignment. AI tutors then redirected the conversation back to WPs and inquired about further assistance in that area. Second, in instances where students provided sufficient context for out-of-scope questions, AI tutors leveraged this information to offer relevant support.

5.2 Students' Feedback

A survey was distributed one week before the end of instruction, with 88% response rate. 79% of students who responded the survey indicated that they have used at least one AI tool in ICS32, with

AI tutors being the most popular choice for these students (58%). 21% of them relied solely on tools like ChatGPT. The use of AI tools in programming classes is likely an inevitable trend, even in the absence of a designated classroom tool. However, unconstrained access to AI tools that provide solutions without guardrails is likely detrimental to student learning.

Then we asked about their overall experience with the AI tutors. This included assessments of AI tutors' perceived helpfulness, advantages, and disadvantages. 79% of students reported the AI tutors had a positive impact on their learning process in ICS32. A summary of the results is as follows:

- (1) The majority students found that AI tutors were helpful with WPs. Specifically, 13% of them "strongly agree," 45% "agree," 30% "neutral," 9% "disagree," and 3% "strongly disagree."
- (2) Even though AI tutors were not designed to assist with other assignments, some students still found they are helpful with 5% "strongly agree," 18% "agree," 45% "neutral," 21% "disagree," and 11% "strongly disagree."

Most students appreciated AI tutors' ability to clearly explain WPs and recommend effective methods. Some valued the step-by-step guidance for problem-solving and its contribution to their understanding of programming concepts. Notably, a significant advantage of AI tutors was that students did not have to worry about being judged because of the questions they asked.

On the down side, most students expressed dissatisfaction with AI tutors' slowness. The implementation we had at the time was, indeed, very slow, and this affected the user experience. We have since fixed this issue, but not in time for ICS32. Another concern raised by students was the occurrence of contradictory responses. An example of this is illustrated in Figure 5, where AI tutor contradicted itself. Some students felt that AI tutors lacked detailed explanations. This could be attributed to the limitation of not incorporating context from past homework assignments.

5.3 Impact on grades

Note: the results presented here stand on a highly complex, multivariate context. Nevertheless, since we had access to the grades of the previous year's course, we make a simple comparative analysis of the grades, to try to identify any statistically significant effects of the AI tutors.

We compared the grade distributions of the cohorts from the Winter 2023 (without AI tutors) and Winter 2024 (with AI tutors) offerings of the class by the same instructor. In this analysis, we compared the distribution of the grades of each WP between the two cohorts, the distribution of the combined WP grade, the grades of the final exam, and the distribution of the final course letter grades.

To compare the continuous distribution of numerical grades of the WPs, we used classical two-sample Kolmogorov-Smirnov tests [18, 31]. We also used Wilcoxon-Mann-Whitney and Anderson-Darling tests, resulting in similar conclusions. The two-sample KS tests indicated that the grade distributions showed no statistically significant difference between the cohorts in WP1, WP2, and WP5, with resulting p-values of 0.132, 0.248, 0.805. The other two WPs, WP3 and WP4, showed statistically significant differences, with p-values of 7.64×10^{-14} and 0.011, respectively. In WP3, the 2024

Grade	2023 cohort		2024 cohort (using AI tutors)	
	Number of students	Percent of class	Number of students	Percent of class
A	158	44%	130	42%
B	97	27%	101	32%
C	41	11%	37	12%
D	13	4%	9	3%
F	51	14%	33	11%

Table 1: Comparison of the final letter grade distribution between the 2023 and 2024 cohorts. AI Tutors were only used by the 2024 cohort.

cohort showed a worse outcome, while in WP4, the 2024 cohort showed a better outcome. As mentioned before, there was a modification in the deadline for WP3 in 2024, which was increased by a week and made it overlap with midterms. This WP had the lowest submission rate among all assignments.

In this course, the student grades of the individual WPs are combined in a single WP grade, accounting for 4.5% of the student's final grade. To calculate this WP grade, the two lowest-graded WPs of each student are dropped, and the grades of the three highest-graded WPs are added together and then rescaled to a maximum of 4.5 points. The comparison of this combined WP grade distribution between the two cohorts shows a statistically significant difference with a p-value of 0.012 and with the 2024 cohort displaying a mode of 0.2% points higher than the 2023 cohort. The distribution of the numerical grades of the final exam showed no statistically significant difference between the two cohorts, with a p-value of 1.000, indicating that the impact of AI tutors in the student grades was reflected only in the WPs.

Table 1 shows the comparison of the final course letter grades between the two cohorts. To test if the observed difference in the discrete letter grades is significant, we perform a Monte-Carlo test by resampling 1000 distributions from the normalized observed grade distribution of each of the two cohorts [e.g. 11], with each distribution being generated using the number of students in the largest cohort (359). Then, we calculate Kolmogorov-Smirnov tests for each pair of the resampled distribution between the cohorts. Afterward, we calculate the mode of the distribution of the test results using the Venter estimator [36]. This test results in a mode of the p-values of 0.414, apparently indicating that the difference in the final course letter grades between the two cohorts is not significant. However, the higher grades (A) dominate the present letter grade distribution, and it is well known in the literature that Kolmogorov-Smirnov tests are weak in distinguishing effects on tails of the tested distributions [29]. Accordingly, we applied the test to grades B and lower, resulting in p-values with a mode of 0.027 that indicates a statistically significant difference at the tails of the grade distribution. These results show that although a small difference was observed in the highest grades of the course between the two cohorts, it is statistically insignificant. They also indicate that a statistically significant difference between the two cohorts is observed at the lower tail of the grade distribution: in the cohort using AI tutors, the fraction of students receiving good grades (i.e. B) increased at a high statistical significance, while the fraction of students passing the course also improved (i.e. we observe a

statistically significant decrease in the F grades). The observed improvement in the number of students passing the course can be partially explained by a possible impact of the AI tutors in the WPs because even if the difference between the combined WP grades is small (0.2 points, or 4.5% higher in the 2024 cohort), that could be enough to make some students move a final letter grade up.

We note, once again, that this data is not enough to draw any firm causation effect between using AI tutors technologies and student grades. We are only analyzing a single course design and only two individual points in time. Moreover, our experiment is not entirely randomized nor performed under the exact same conditions, as the cohorts of different years are formed by students who may have been educated differently prior to the ICS32 course. Thus, multiple hidden factors may have impacted the grade distributions. Nevertheless, even if further studies are necessary for any causal inference analysis, the current data can suggest that AI tutors could have an impact on student grades that can be uneven among high-achieving students and low-achieving students.

6 Conclusion

This paper introduces RAGMan, an LLM-powered tutoring system designed to support entry-level programming students with homework assignments. AI tutors build on top of RAGMan to offer students a conversational interface to help them with problem-solving without directly providing solutions.

We conducted a manual analysis to evaluate the quality of AI tutors' responses. The results show that 93.5% were categorized as good, meaning they correctly and effectively addressed student inquiries. We also distributed a survey to assess student experiences using AI tutors. Over 78% of students who have used AI tutors reported that the tutors facilitated their learning in class, which indicates that regardless of any real impact on the course outcomes, students felt that they are learning more when they are assisted by such technology. The survey results also show that students appreciated the AI tutors' ability to provide helpful study advice in a judgment-free environment where they felt comfortable asking questions. Efforts to improve AI tutors will focus on minimizing the generation of contradictory responses.

Finally, we compared the grade distributions of a course cohort that used AI tutors and another that did not, offered in two different years. This comparison shows a statistically significant increase in the number of students approved in the course and an increase in the number of students receiving middle grades (i.e., B). This data suggests, although it does not demonstrate, that AI tutors can positively impact student success and provide important help, especially to students who would be struggling in challenging courses.

References

- [1] Hussam Alkaiissi and Samy I McFarlane. 2023. Artificial hallucinations in Chat-GPT: implications in scientific writing. *Cureus* 15, 2 (2023).
- [2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Ben-netot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion* 58 (2020), 82–115.
- [3] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*

- V. 1 (Toronto, ON, Canada) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 500–506. <https://doi.org/10.1145/3545945.3569759>
- [4] Som Biswas. 2023. Role of ChatGPT in Computer Programming.: ChatGPT in Computer Programming. *Mesopotamian Journal of Computer Science* 2023 (2023), 8–16.
 - [5] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 1136–1142.
 - [6] Eva Durall and Evangelos Kapros. 2020. Co-design for a competency self-assessment chatbot and survey in science education. In *Learning and Collaboration Technologies. Human and Technology Ecosystems: 7th International Conference, LCT 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II* 22. Springer, 13–24.
 - [7] James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, James Prather, and Brett A. Becker. 2023. My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In *Proceedings of the 25th Australasian Computing Education Conference (Melbourne, VIC, Australia) (ACE '23)*. Association for Computing Machinery, New York, NY, USA, 97–104. <https://doi.org/10.1145/3576123.3576134>
 - [8] Philip J. Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (Denver, Colorado, USA) (SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 579–584. <https://doi.org/10.1145/2445196.2445368>
 - [9] Steven Halim. 2015. Visualgo—visualising data structures and algorithms through animation. *Olympiads in informatics* 9 (2015), 243–245.
 - [10] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556* (2022).
 - [11] Adery C. A. Hope. 1968. A Simplified Monte Carlo Significance Test Procedure. *Journal of the Royal Statistical Society. Series B (Methodological)* 30, 3 (1968), 582–598. <http://www.jstor.org/stable/2984263>
 - [12] Sharin Jacob, Tamara Tate, and Mark Warschauer. 2023. Emergent AI-assisted discourse: Case study of a second language writer authoring with ChatGPT. *arXiv preprint arXiv:2310.10903* (2023).
 - [13] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.
 - [14] Enkelejd Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences* 103 (2023), 102274.
 - [15] Enkelejd Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103 (2023), 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
 - [16] Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. 2024. Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A* 382, 2270 (2024), 20230254.
 - [17] Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Z Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. *arXiv preprint arXiv:2401.11314* (2024).
 - [18] Andrey Nikolaevich Kolmogorov. 1933. Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari* 4 (1933), 83–91. <https://cir.nii.ac.jp/crid/1571135650766370304>
 - [19] Mohammad Amin Kuhail, Nazik Alturki, Salwa Alramlawi, and Kholood Alhejori. 2023. Interacting with educational chatbots: A systematic review. *Education and Information Technologies* 28, 1 (2023), 973–1018.
 - [20] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing code explanations created by students and large language models. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 124–130.
 - [21] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A Becker. 2023. Using large language models to enhance programming error messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 563–569.
 - [22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
 - [23] Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HalaEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*. <https://openreview.net/forum?id=bxsrkzSnq>
 - [24] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2023. Codehelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. 1–11.
 - [25] Jionghao Lin, Ashish Gurung, Danielle R Thomas, Eason Chen, Conrad Borchers, Shivang Gupta, Kenneth R Koedinger, et al. 2024. Improving Assessment of Tutoring Practices using Retrieval-Augmented Generation. *arXiv preprint arXiv:2402.14594* (2024).
 - [26] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J. Malan. 2024. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (<conf-loc>, <city>Portland</city>, <state>OR</state>, <country>USA</country>, </conf-loc>) (SIGCSE 2024). Association for Computing Machinery, New York, NY, USA, 750–756. <https://doi.org/10.1145/3626252.3630938>
 - [27] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 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*. 931–937.
 - [28] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language Models of Code are Few-Shot Commonsense Learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 1384–1403. <https://doi.org/10.18653/v1/2022.emnlp-main.90>
 - [29] David M. Mason and John H. Schuenemeyer. 1983. A Modified Kolmogorov-Smirnov Test Sensitive to Tail Alternatives. *The Annals of Statistics* 11, 3 (1983), 933–946. <http://www.jstor.org/stable/2240655>
 - [30] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446* (2021).
 - [31] Nikolai Vasilyevich Smirnov. 1939. Estimate of deviation between empirical distribution functions in two independent samples. *Bull. Moscow Univ.* 2 (1939), 3–16.
 - [32] Jacob Steiss, Tamara Tate, Steve Graham, Jazmin Cruz, Michael Hebert, Jiali Wang, Youngsun Moon, Waverly Tseng, Mark Warschauer, and Carol Booth Olson. 2024. Comparing the quality of human and ChatGPT feedback of students' writing. *Learning and Instruction* 91 (2024), 101894. <https://doi.org/10.1016/j.learninstruc.2024.101894>
 - [33] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulkshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lambda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).
 - [34] Ahmed Tlili, Boulus Shehata, Michael Agyemang Adarkwah, Aras Bozkurt, Daniel T Hickey, Ronghuai Huang, and Brighter Agyemang. 2023. What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education. *Smart Learning Environments* 10, 1 (2023), 15.
 - [35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
 - [36] J. H. Venter. 1967. On Estimation of the Mode. *The Annals of Mathematical Statistics* 38, 5 (1967), 1446–1455. <http://www.jstor.org/stable/2238960>
 - [37] Hongyan Wang, WeiZhen Wu, Zhi Dou, Liangliang He, and Liqiang Yang. 2023. Performance and exploration of ChatGPT in medical examination, records and education in Chinese: Pave the way for medical AI. *International Journal of Medical Informatics* 177 (2023), 105173.
 - [38] Mark Warschauer, Waverly Tseng, Soobin Yim, Thomas Webster, Sharin Jacob, Qian Du, and Tamara Tate. 2023. The affordances and contradictions of AI-generated text for second language writers. *Available at SSRN* (2023).
 - [39] Jason Lee Weber, Barbara Martinez Neda, Kitana Carbajal Juarez, Jennifer Wong-Ma, Sergio Gago-Masague, and Hadar Ziv. 2024. Measuring CS Student Attitudes Toward Large Language Models. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2* (Portland, OR, USA) (SIGCSE 2024). Association for Computing Machinery, New York, NY, USA, 1846–1847. <https://doi.org/10.1145/3626253.3635604>