Using LLMs for Support in High School Programming Education

by

Thom van der Velden

Thesis Advisor:	M. Specht
Daily Co-Supervisor:	M. Valle Torre
Project Duration:	August, 2024 - May, 2025
Faculty:	Faculty of Computer Science and Engineering, Delft

TU Delft

Preface

Writing this thesis gave me the chance to combine my role as a teacher with my interest in educational technology, especially the growing use of Large Language Models (LLMs) in classrooms.

Over the past year, I worked together with a class of high school students as they learned Python for the first time. Along the way, we explored how LLMs could help them understand programming concepts, fix bugs, and ask questions in real time. But I also wanted to find out: does using these tools really help students learn better? Or could it hurt their learning?

This project would not have been possible without the students who participated and asked so many interesting (and sometimes surprising) questions. I am also very thankful to my daily co-supervisor, Manuel Valle Torre and my thesis advisor, Dr. Marcus Specht, for all their feedback, support, ideas, and most of all, the fun times.

Thanks also to my friends and family who supported me throughout the thesis. It would not have been possible without them.

Working on this thesis showed me that if we want AI tools to really help students, we need to think carefully about how we design and use them. I hope this research adds something useful to that conversation.

Part of this thesis was submitted and is currently under review for the *European Conference on Technology Enhanced Learning* (EC-TEL).

Thom van der Velden
Delft, April 2025

Summary

Introductory programming education poses unique challenges; beginners often struggle with abstract concepts, computational procedures, and opaque error messages, often leading to frustration and reliance on memorization. Large Language Models (LLMs) offer scalable, personalized support but risk hindering independent problem-solving through overreliance. To explore this tension between support and tension, we investigated the interactions of 18 high school students in JELAI, a programming environment with an LLM chatbot, over a 12-week Python course. We analysed interaction frequency, the impact of instrumental (adaptive: seeking understanding) versus executive (non-adaptive: seeking solutions) query types on exam performance, and the effectiveness of an intervention promoting instrumental strategies. We also explored the feasibility of automatic query classification using a BERT model, given its potential for scalable analysis of student-LLM interactions. Higher LLM interaction frequency correlated negatively with exam scores, particularly for students with frequent executive queries. As expected, executive help-seeking strongly predicted lower grades. Surprisingly, however, frequent instrumental comprehension queries also correlated negatively with performance, likely indicating that high interaction frequency signals persistent difficulty. A mid-course intervention successfully reduced the proportion of executive queries and fostered more effective interaction workflows, coinciding with a small average grade increase. Automatic classification successfully distinguished instrumental and executive query types (82.5% accuracy). This study highlights that LLM integration requires pedagogical strategies and system designs that actively guide students towards instrumental help-seeking and function as learning partners, rather than mere solution providers.

Contents

List of Figures

List of Tables

1 Introduction

Teaching programming to high school students comes with unique challenges. For example, many students initially struggle to debug cryptic error messages and grasp abstract concepts such as loops and conditionals, while others might overly depend on memorizing syntax rather than understanding the logic behind it. Persistent challenges like these have been widely documented [1].

Large Language Models (LLMs) offer a promising solution for addressing these challenges. By providing personalized, real-time help [2], LLMs can help students overcome syntax-related obstacles and focus their efforts on understanding core programming concepts and logic. LLMs can take care of the tedious parts of programming, for example, debugging syntax errors [3]. This lets students focus on bigger-picture concepts and develop problem-solving skills that are useful beyond just programming. In addition, the use of LLMs in education opens up new ways of designing courses that are tailored to each student [4].

Despite this promise, there are risks associated with the use of LLMs in education. Over-reliance on these tools may hinder the development of independent problem-solving skills and reduce students' familiarity with programming syntax. This lack of foundational understanding could hinder students from effectively debugging and comprehending their own code. Zhai et al. (2024) systematically reviewed the effects of over-reliance on AI systems and how such dependence can weaken essential cognitive skills, including decision-making, critical thinking, and analytical reasoning. Their findings emphasize the need to guide the usage of AI tools to ensure that students critically evaluate AI-generated output [5]. This is even more important because LLMs also face limitations, for example, in addressing complex logical errors or providing context-aware guidance.

Existing research on the use of LLMs in education focuses more on university-level learners, with limited exploration of their applicability to younger audiences. High school students, with their limited prior experience and distinct stages of cognitive development, represent a demographic that requires investigation. Furthermore, many of these studies do not have access to the actual questions that students are asking the LLM.

1.1. Problem Statement

This thesis seeks to address this research gap by examining the role of LLMs in supporting programming education for Dutch high school students. Through the integration of LLM-assisted environments within a structured programming course, this research explores how different types of LLM interactions, such as requesting explanations versus seeking solutions from the LLM, impact learning outcomes. By analyzing student behavior, performance, and question patterns within this context, the study aims to provide insights into the effective use of LLMs in programming education.

In doing so, this research not only addresses a critical gap in the literature but also offers practical implications for educators seeking to integrate LLMs into high school programming curricula. The findings will help develop strategies for maximizing the potential of LLMs while mitigating risks, ensuring

that these tools complement rather than replace the essential learning processes in (programming) education.

1.2. Research Questions

This thesis investigates the impact of LLM usage on the learning outcomes of high school students engaged in programming. The central research question that guides this study is the following:

How does the use of an LLM affect the learning outcomes of high school students who are learning to program?

To address this overarching question, the following sub-questions will be explored:

1. How does the frequency of student interactions with an LLM correlate with their programming performance?
2. How do different types of student interactions with an LLM (e.g., seeking explanations vs. requesting direct solutions) impact learning outcomes?
   (a) Can student queries to an LLM be automatically categorized, enabling adaptive educational responses?
3. How does an intervention aimed at guiding LLM usage influence students' help-seeking and learning behaviours?

1.3. Thesis Outline

The remainder of this thesis is structured as follows: Chapter 2 reviews the relevant literature, covering six key areas: challenges in programming education, learning analytics, the role of LLMs in education, self-regulated learning and help-seeking behavior, interaction strategies with LLMs, and ethical considerations.

Chapters 3 and 4 are about the research methodology, which consists of two main components: the design of the experiment and the development of the classifier used to categorize student queries.

Chapter 5 presents the findings of this study, followed by Chapter 6, which discusses the implications of the results, potential limitations, and directions for future research. Finally, Chapter 7 summarizes the key insights from this research and provides final reflections on the role of LLMs in high school programming education.

2

Related Work

In this chapter, the relevant literature is reviewed to provide a theoretical and empirical foundation for this study. This chapter is organized into six different sections that reflect key themes and research areas relevant to this thesis.

2.1. Challenges in (High School) Programming Education

Learning to program is widely known to be difficult for beginners, whether in high school or in introductory university courses. Studies of first-year university students around the world report high failure and dropout rates in introductory programming classes [6]. Learning to program requires cognitive abstraction skills, algorithmic problem-solving, and logical reasoning, which many beginners find challenging [7]. Part of the problem is how programming is taught. The materials often do not match how students learn best, and many programming languages focus on being useful for professionals instead of being easy to learn [1]. Unsurprisingly, this leads to frustration and therefore high dropout rates.

Another challenge in programming education are misconceptions. Qian and Lehman highlight that students often struggle with core programming concepts, such as loops, variables, and conditionals, because they develop inaccurate mental models of how these constructs work. These misunderstandings lead to persistent errors that are difficult to correct without explicit intervention [8]. Similarly, Alqadi and Maletic found that debugging poses a significant challenge for novice programmers, as many students lack effective strategies for locating and fixing errors. They often rely on trial-and-error approaches rather than systematic debugging techniques [9]. Without a structured way to identify and resolve mistakes, students become frustrated and disengaged. Addressing these challenges requires a combination of proactive teaching strategies—such as conceptual scaffolding and targeted debugging instruction—to help students develop effective debugging techniques, strengthen their problem-solving skills, and gain a deeper understanding of programming concepts.

Research indicates that computer science (CS) teachers at various school levels often lack adequate training and support resources, which undermines the quality of CS education. For example, Ni and Guzdial observed that the fast expansion of K–12 CS education has led to a high demand for qualified teachers, but many instructors enter these roles without proper training or certification [10]. This shortage of well-prepared teachers can be seen at the primary school level as well. Salleh Hudin et al. highlight challenges faced by primary school teachers in teaching coding, attributing these difficulties to a lack of sufficiently skilled (trained) instructors and inadequate support (e.g. limited resources and minimal governmental aid) [11]. Taken together, these findings show a consistent problem across educational levels: many CS teachers are not fully prepared or do not have the resources to teach effectively.

When we look at all these challenges: persistent misconceptions, limited teacher preparation, and the need for better student support, it becomes clear that effective solutions must offer real-time assistance, address gaps in teacher expertise, and help students build accurate mental models. Large Language Models (LLMs) show promise in meeting these needs by providing personalized feedback and adaptive

guidance. In the following sections, we will explore how LLMs, combined with Learning Analytics, can help address these challenges and enhance programming education.

2.2. Learning Analytics in Programming Education

Learning Analytics (LA) involves the measurement, collection, analysis, and reporting of educational data to better understand and optimize learning processes [12]. It draws from fields like education, data science, and human-computer interaction, leveraging digital data to enhance student outcomes. Initially, LA was used to identify at-risk students, but more recent approaches focus on proactive interventions, such as providing real-time feedback and personalized learning support [13].

In programming education, Learning Analytics enables the analysis of rich data generated as students write and debug code, offering insights beyond traditional assessment scores. Instead of just focusing on final grades, research suggests that examining detailed process data—like how students code, solve problems, and debug—can provide deeper insights into their learning [14]. This cognitive level analysis allows educators to move beyond correctness-based assessments and study how students arrive at solutions, revealing key learning behaviors. As a result, LA in programming education plays a critical role in enabling real-time feedback and adaptive learning interventions.

Real-time data analysis has enabled instructors to provide immediate feedback and timely interventions in programming courses. By continuously collecting student coding data (e.g. compile events and error logs) at short intervals, educators can spot when a learner is struggling. This immediate insight is critical, as research suggests that analyzing the ongoing coding process (rather than just final code submissions) can better predict student success, highlighting the value of continuous monitoring [15].

Many modern learning environments leverage real-time feedback mechanisms to keep students on track. For example, Fu et al. developed a system called LAPLE that displays a live dashboard of student progress and groups students by knowledge level, helping instructors quickly identify who is at risk and adjust instruction on the fly. At the same time, LAPLE provides students with immediate, in-editor support by detecting syntax errors and offering recommendations on how to fix them, so learners can iterate and learn from mistakes in real time [16]. Such instant feedback loops have been used to maintain student engagement and reduce frustration, therefore enhancing the programming learning experience.

Pereira et al. further contribute to this by introducing CodeBench, an Online Judge system that provides fine-grained learning analytics in programming courses. Their study, conducted over three years with data from 2058 students, identified clear behavioral patterns among effective, average, and ineffective learners. Using machine learning techniques such as k-means clustering and association rule mining, they found that early behavioral indicators—including syntax errors, debugging patterns, and time spent coding—could accurately predict student success or failure [17]. The study emphasizes the importance of real-time analytics and targeted interventions, allowing educators to guide struggling students based on data-driven insights.

Marwan et al. integrated an adaptive real-time feedback system into a programming environment and observed substantial gains in student focus and participation, along with a marked increase in students' intention to persist in computer science programs [18]. Such analytics-driven, adaptive feedback keeps learners actively involved in coding tasks and helps them identify and correct mistakes quickly, which strengthens their problem-solving skills.

Overall, learning analytics offers valuable insights into programming education, however there are still opportunities to enhance its real-time applications and deepen our understanding of the cognitive processes behind coding. Future research should aim to integrate LA tools directly into programming environments to provide adaptive, personalized feedback that boosts student engagement with coding. Large language models could play a significant role in this and in the following section we will explore what LLMs are and how they are already being integrated into (programming) education.

2.3. LLMs and Programming Education

Large Language Models (LLMs) are advanced AI systems designed to process and generate human-like text. These models typically consist of deep neural networks with billions of parameters, and they

are trained on huge datasets of human language (e.g. text from books, articles, and websites). [19]. By analyzing statistical patterns in all this data, an LLM learns to predict the most likely next word or sequence of words given a context. Through this mechanism, LLMs can produce contextually relevant sentences, capturing many nuances of human language and knowledge [20].

Over the past decade, LLMs have progressed very fast, driven by a few key breakthroughs in model architecture and training techniques. One of the most important developments was the introduction of the Transformer architecture in 2017, a self-attention-based neural network that enabled training much larger and more generalized language models [21]. This innovation led to models like BERT [22] and GPT [23] in 2018, which demonstrated that pre-training on unlabeled text combined with fine-tuning for specific tasks yields far superior language understanding and generation than earlier task-specific approaches. Researchers soon found that scaling up model size and training data further improves performance, culminating in extremely large models such as GPT-3 with 175 billion parameters [24]. More recently, GPT-4 has built on these advances by employing even larger architectures and refined training methods to achieve state-of-the-art performance in various language tasks [25].

These recent advancements offer potential solutions for providing scalable, personalized support in programming education [26]. LLMs are being explored across various areas of education, from tutoring and writing assistance to personalized feedback and language learning. Because they can understand questions and generate detailed explanations, LLM-based systems have been deployed as virtual tutors that provide one-on-one guidance to students, answer questions, and adapt to individual learning needs [27]. Similarly, in language learning, conversational AI tutors allow learners to practice a new language interactively. Recent research showed that a chatbot powered by an LLM significantly boosted second-language vocabulary acquisition and retention compared to traditional study methods [28].

Over the last years, researchers have developed specialized LLM-based systems for programming education, aiming to provide help that supports learning. Some prominent examples are: CodeTutor: A chatbot using system prompts for educational responses; its interface is similar to that of ChatGPT and allows students to rate feedback at both the message and conversation levels [29]. CodeAid: An AI tutor providing targeted support guiding the student via specific feature selection while avoiding direct solutions [30]. And CodeHelp: A web interface using multiple prompts to ensure an appropriate response from the LLM, which also avoids giving away solutions [31]. A characteristic of these systems is that they were not intended to research student interactions as they work in the coding environment. The system used in this study, JELAI, addresses this by integrating the LLM chatbot directly into its notebook environment, similar to modern development interfaces (like Github Copilot), potentially making interaction more intuitive and realistic.

The application of LLMs in programming education presents opportunities, but also challenges. Jošt et al. found that while LLMs can provide valuable assistance with debugging, syntax correction, and logical error resolution, over-reliance on these tools can negatively impact students' ability to independently solve programming problems. Their study revealed that students who frequently used LLMs to generate and debug code tended to perform worse in programming assessments, showing the importance of balancing AI assistance with fundamental problem solving skills [32].

Zhai et al.take this discussion further by exploring the broader cognitive risks associated with over-reliance on AI dialogue systems. They emphasize that excessive reliance on LLMs can weaken critical thinking, decision-making, and analytical reasoning skills. Students may become too dependent on AI-generated output without critically evaluating their accuracy or validity. This brings us to another concern about LLMs: hallucinations (the generation of misleading or false information). If students cannot determine whether an LLM is telling the truth, hallucinations can be very detrimental to learning. Ethical concerns, such as algorithmic bias, and privacy issues, further complicate the integration of these tools into education. Zhai et al. recommend strategies to mitigate all of these risks, such as teaching students to critically evaluate AI-generated content and fostering a balanced approach to AI use [5].

Interestingly, Zhai et al. also note that while LLMs pose challenges, they offer significant benefits when used responsibly. For example, students can use these tools to seek additional explanations, gain insight, and improve learning efficiency without replacing the need for independent problem solving. Their findings reinforce the idea that LLMs should complement traditional teaching methods rather

than serve as standalone solutions.

Lieb and Goel extend this discussion by evaluating the impact of LLM-based tutoring chatbots through their study of NewtBot, a GPT-3.5-powered chatbot designed to assist secondary school physics students. Their research looked at three chatbot configurations: a general-purpose Baseline model, a Tutor model tailored for structured guidance, and a Feedback model offering problem-specific explanations. Their findings showed that students preferred structured tutoring over generic AI responses, reinforcing the importance of scaffolded AI interactions in education. Although students appreciated NewtBot's efficiency, they remained skeptical about whether AI tutoring could significantly improve their grades. These insights highlight the need to thoughtfully design LLM-based tutoring systems to enhance engagement while avoiding over-reliance on AI [33].

Ultimately, while LLMs offer exciting opportunities to enhance programming education, they also come with risks. Striking the right balance between leveraging these tools for support and ensuring students develop independent skills is crucial. As the use of LLMs continues to expand, educators must adopt thoughtful strategies to maximize their benefits while minimizing potential drawbacks. More empirical research is needed to understand the effects and underlying mechanisms of LLM use in education [34]. In the following section, we look into self-regulated learning (SRL) and help-seeking behavior, exploring how these frameworks can guide responsible LLM use and promote deeper and more independent learning.

2.4. Self-Regulated Learning and Help-Seeking

Self-regulated learning (SRL) refers to students' ability to actively control their learning process through planning, monitoring, and reflection [35]. A crucial component of SRL is help-seeking behavior, where students recognize when they need assistance and strategically seek support. Effective help-seeking is an important metacognitive skill that contributes to deeper learning and problem-solving abilities [36]. However, novices often lack the metacognitive skills or confidence to ask for appropriate assistance [37, 38, 39].

In traditional classroom settings, students can seek help from teachers or peers, but in interactive learning environments (ILEs), such as AI-driven tutoring systems or LLM-based programming assistants, help-seeking takes on a different dynamic. Studies show that students often misuse or underuse on-demand help in these environments [40]. Some students rely heavily on AI-generated solutions, bypassing critical thinking, while others avoid using help even when they struggle. These ineffective help-seeking behaviors can decrease actual learning and create a dependency on external tools.

Research on LLM-powered programming assistants further illustrates this. Sheese et al. analyzed over 2,500 student queries to an LLM-based tutor, CodeHelp, during a semester-long programming course. Their findings revealed that students primarily sought immediate debugging and implementation assistance, with only 8% of queries focused on conceptual understanding. Moreover, many students submitted low-effort queries, either copy-pasting assignment instructions or failing to describe their expected program behavior. This help-seeking behavior reduced the quality of AI-generated responses and only helped short-term problem-solving instead of helping with deeper learning [31].

The need to develop metacognitive awareness in help-seeking has been a major focus in research on Intelligent Tutoring Systems (ITS). According to Aleven et al., asking for help the right way is not something people are born knowing—it is a skill that needs to be taught. They emphasize that effective help should be context-aware, adaptive, and structured to promote active engagement. When help is misused, either through over-reliance on direct solutions or complete avoidance, learning suffers. This is particularly relevant for LLM-based programming education, where students often request complete code solutions rather than engaging in step-by-step reasoning [36].

One approach to improving help-seeking behavior is metacognitive feedback, which provides students with real-time guidance on when and how to seek help effectively. Roll et al. implemented an intelligent Help Tutor within a geometry-based ITS and found that students who received immediate feedback on their help-seeking errors improved their help-seeking strategies over time [41]. Their study showed that when students were asked to reflect before requesting additional hints, they were less likely to overuse bottom-out hints (fully worked-out answers) and more likely to engage in self-explanation. However,

simply providing metacognitive feedback was not enough; students needed additional guidance and explicit instruction to transfer these skills beyond the immediate system.

This aligns with the ICAP framework [42], which categorizes learning interactions as Passive, Active, Constructive, and Interactive, with deeper learning occurring when students actively engage with instructional content. When students construct knowledge by engaging with scaffolded help prompts, rather than receiving direct answers, they are more likely to internalize concepts and develop independence.

We could further explore help-seeking strategies by dividing them into two different categories: instrumental and executive help-seeking. Instrumental help-seeking involves students actively engaging in the problem-solving process, requesting hints or partial guidance that supports independent learning. Instrumental help-seeking is strongly associated with better retention and skill transfer [37]. In contrast, executive help-seeking is characterized by requests for complete solutions or direct answers, hindering long-term learning [43].

A key challenge in LLM-assisted programming education is helping students develop strong help-seeking strategies that support long-term learning and independence. We want students to ask instrumental-type questions [44]. Over-reliance on AI assistance, particularly using LLMs excessively for direct solutions rather than explanations, can impair learning [45], especially for novice learners [46]. AI tutoring systems can address this by guiding student interactions, offering step-by-step hints, and encouraging reflection on mistakes, reducing the risk of over-reliance on AI. In the following section, we explore how different interaction strategies with LLMs influence learning outcomes and help-seeking behavior.

2.5. Interaction Strategies with LLMs

This section analyzes how different interaction strategies (e.g., asking for explanations versus requesting solutions) affect learning outcomes.

Kumar et al. explored the impact of LLM guidance and interaction strategies on student learning, focusing on math education. Their study found that customized LLM-generated explanations significantly improved learning outcomes, particularly when students attempted problems first before seeking AI assistance. This highlights the importance of sequencing in learning, where engaging with a problem before consulting an LLM leads to better retention and understanding. Additionally, their findings show that tailored explanations promote conceptual understanding, making LLMs valuable as scalable, adaptive tutors. While the study was conducted in a math context and with older learners, its insights, such as the ”solve-first-then-consult” approach, are highly relevant to programming education. These strategies suggest that structured LLM interactions can enhance problem-solving skills when implemented thoughtfully, reinforcing the need for guided AI use in educational settings [47].

Kudina et al. examined how LLMs can support proleptic reasoning, a method where students anticipate and engage with counterarguments. Their study found that structured interactions with LLMs foster deeper analytical thinking and critical reasoning skills. Through iterative dialogue, students refine their arguments, challenge assumptions, and strengthen their conceptual understanding. This aligns with broader educational goals of active learning, where AI is used not merely as an information source but as a tool that encourages students to articulate, reflect on, and improve their reasoning. While the study primarily focuses on argumentation, its findings suggest that structured prompting and guided interactions with LLMs could also be applied to programming education, helping students debug code more effectively, understand underlying logic, and refine problem-solving approaches [48].

Maiti & Goel provide further insights by analyzing student interactions with Jill Watson, an LLM-powered virtual teaching assistant deployed across multiple courses. Their study found that students tend to use AI primarily for factual recall, but structured AI interventions can encourage higher-order thinking and problem-solving skills. The complexity of student queries varied significantly based on course structure, with STEM courses generating more analysis-based questions while writing-intensive courses relied on AI for basic information retrieval. Student engagement with AI spiked before major exams, suggesting that LLM-powered tutors are often used reactively rather than as ongoing learning tools. These findings reinforce the need for active guidance in LLM-based programming education, ensuring that students develop self-regulated learning habits rather than resorting to AI as a shortcut [49].

Salminen et al. further highlight engagement challenges in AI tutoring systems by studying Cipherbot, an LLM-based chatbot used in a classroom setting. Their study found that 82.5% of student queries were answered successfully, demonstrating the scalability of AI tutors, but also revealed declining student engagement over time. The chatbot was often used by a small group of "power users", while others engaged minimally, suggesting that without structured guidance, some students may over-rely on AI while others underutilize it. These insights reinforce the need for scaffolded AI interactions, ensuring that students ask meaningful questions, critically evaluate AI-generated responses, and use LLMs as a complement rather than a replacement for problem-solving [50].

Together, these studies suggest that LLMs can significantly enhance programming education when used strategically, but unguided interactions can lead to dependency or superficial learning. The key challenge is designing interventions that encourage students to engage meaningfully with AI tutors, rather than relying on them for quick solutions. However, beyond pedagogical considerations, the integration of LLMs also raises critical ethical questions. In the next section, we delve into these ethical considerations and discuss how responsible AI usage can be promoted in educational settings.

2.6. Ethical Considerations

Integrating AI tutors into education raises significant ethical considerations. A primary concern is algorithmic bias in AI-generated feedback or content. Because LLMs learn from vast datasets of human-produced text, they can reproduce stereotypes or biased viewpoints, leading to unequal or even toxic responses [51]. This poses risks for fairness and equity – for instance, if an AI tutor consistently misunderstands certain dialects or cultural references, students from those backgrounds could have a disadvantage.

Another concern is privacy: using AI in the classroom may involve collecting or sharing student data (e.g. learning profiles or query logs). Without strict data protection measures, this information could be misused, leading to breaches of student confidentiality or unauthorized access to sensitive records. Regulations such as FERPA in the United States and GDPR in Europe mandate strict safeguards for student data privacy, yet the increasing reliance on AI tools raises questions about how these policies are enforced in educational settings [52].

Large Language Models can sometimes produce correct looking, but incorrect information, something known as hallucinations in AI. Hallucinations occur when an AI-generated response contains false or misleading details presented as fact [53]. The danger in education is that students might trust these responses and learn from them. If learners accept AI-generated hallucinations as truth, they can develop misconceptions that undermine their understanding of key concepts. Such hallucinations may increase differences in student performance, benefiting those who can recognize these hallucinations while disadvantaging those who cannot. This risk is not only theoretical: one study found that college students readily trusted ChatGPT's answers and even submitted them as homework, despite the responses containing multiple factual errors [54]. These cases demonstrate how blindly trusting AI-generated content can result in misinformation and hinder meaningful learning.

Another ethical consideration is that a large dependence on AI tutors or assistants can hinder developing students' critical thinking and problem-solving abilities. Frequent dependence on AI-generated solutions can prevent students from fully engaging with complex problems, limiting their ability to develop deep conceptual understanding. Empirical research supports this concern: a recent study found a significant negative correlation between frequent AI tool use and students' critical thinking performance, suggesting that heavy reliance leads to increased cognitive offloading (outsourcing of mental effort) and reduced analytical engagement [55]. In STEM fields, where hands-on problem-solving and complex reasoning are very important, such over-reliance on AI may hinder students' practice and slow down the development of their analytical skills.

Addressing these AI-related ethical concerns requires a multi-faceted strategy. One approach is to improve AI literacy among both students and educators. Experts stress that teachers and learners must develop new competencies to understand how AI systems work and recognize their limitations [56]. Such AI literacy programs can get users to critically evaluate AI responses rather than accept them without thinking. In addition, promoting transparency in AI tools is crucial – for instance, designers can implement features that explain an AI's reasoning or provide source references, helping learners verify

the information [57]. Equally important is integrating AI in a teacher-guided manner. Rather than allowing AI tutors to operate in isolation, educators should supervise and monitor AI use in the classroom. By combining AI literacy education, transparency measures, and strong teacher oversight, schools can harness the benefits of AI tutors while keeping accuracy, accountability, and student-centered learning.

3

Methodology Experiment

Building upon the literature review in Chapter 2, this chapter outlines the methodology used in this study to investigate the role of LLMs in supporting programming education for high school students.

In the first section, we examine the research design and explain our mixed-methods approach. This is followed by a description of the participants, including ethical considerations. Subsequently, the implementation of the LLM-assisted programming environment, JELAI, is explained. Finally, the chapter discusses the structure of the course, the data collection methods, and the analysis techniques used to understand the impact of LLMs on learning outcomes and student experiences.

3.1. Research Design

3.1.1. Mixed-Methods Approach

This study employs a mixed-methods research design to gain a comprehensive understanding on the impact of LLMs on high school students' programming learning experiences. The quantitative component of the study involves analyzing student interactions with the LLM, categorizing their queries, and correlating patterns with learning outcomes. The qualitative component consists of examining students' behaviors and perceptions of LLM assistance through surveys and interviews.

3.1.2. Experimental Design

The students participated in a structured programming course where they learned the basics of programming. During this course, students could seek assistance from an LLM called Juno, while they were doing exercises.

The experiment consisted of multiple phases:

1. **Survey** – At the beginning of the course, students completed a survey to measure their initial programming knowledge and LLM usage.
2. **LLM-Assisted Learning 1** – In the first part of the course, the students were free to use the AI in any way. Their interactions with the LLM were logged and categorized.
3. **Midterm Exam** - Midterm exam about the very basics of Python.
4. **Intervention** – After the midterm exam, we introduced strategies to optimize students' use of the LLM and provided individualized feedback on areas for improvement.
5. **LLM-Assisted Learning 2** – In the second part of the course, additional programming content was introduced, with the expectation that students would apply the feedback received to use the LLM more effectively.
6. **Endterm Exam** - Endterm exam about all the course materials.
7. **Interviews** – After the course, students participated in interviews to capture their perceptions of LLM assistance and JELAI.

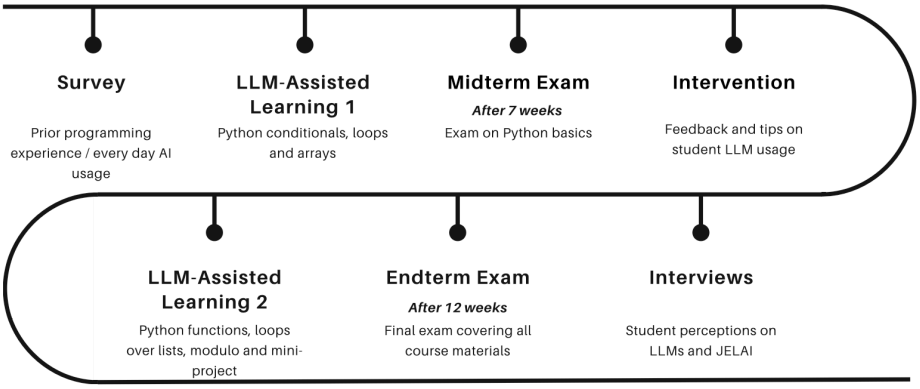

Figure 3.1: Experimental design

By combining quantitative performance analysis, with qualitative insights, this experimental design enables us to clearly evaluate the benefits and challenges of LLM-assisted programming education.

3.2. Participants

3.2.1. Recruitment and Demographics

Participants were recruited from an *Informatica* class at a Dutch high school. *Informatica* is an elective class where students are introduced to computer science concepts and programming. A total of 18 students, aged between 15 and 18 years, participated in the study. The majority of the participants were male, but we did not look into the differences in behaviour by gender.

At the start of the course, an initial survey was given to the students to gather demographic data and assess the students' prior programming experience and attitudes toward learning programming. These responses provided valuable information on the baseline of the programming experience of these students and the attitudes of the participants, which is essential to understand how LLM-assisted learning impacts their programming education.

3.2.2. Ethical considerations

The study was conducted in a classroom setting where I, the writer of this thesis, am the teacher, therefore we had to be very careful with all of the ethical risks involved. Before starting the actual experiment, we prepared and submitted a Human Research Ethics Committee (HREC) form (see Appendix B) outlining our research objectives, the data we wanted to collect, and the risks involved with this. Alongside the HREC submission, a comprehensive data management plan (DMP) (see Appendix A) was created to ensure that all student data was handled securely and in compliance with applicable data protection standards.

Given that the study involved capturing student interactions with the LLM and gathering additional survey and interview data, obtaining informed consent (see appendix A) from the students was crucial. As most of the participants were underage, we obtained consent not only from the students, but also from their parents. Detailed consent forms, which described the study procedures, data usage and confidentiality measures, were distributed and signed by both students and their parents. Only after receiving full approval from the ethics committee and all necessary consents did we start the experiment in the classroom.

This elaborate ethical process ensured that the study upheld high standards of participant protection and transparency, while still allowing us to investigate the impact of LLM-assisted programming education.

3.3. Course Structure

The programming course was organized into two parts, designed to progressively build the students' proficiency in Python programming. In the first part of the course, held from early October through early December, students were introduced to the very basics of programming. In the second part of the course we went a bit deeper and we ended with a small project in which students had to create a mini-game in the console.

Date	**Topic**
7 Oct	Introduction to system and chatbot / print
11 Oct	Variables and input
14 Oct	Math / Integers / data types
18 Oct	Math / Integers / data types
Autumn break	
4 Nov	Rehearsal + intro booleans
8 Nov	Booleans / simple conditional
11 Nov	Conditionals if/else
15 Nov	Conditionals elif
18 Nov	While loops
22 Nov	While loops
25 Nov	For loops
29 Nov	Arrays
2 Dec	Arrays
6 Dec	Practice exam / questions
9 Dec	Practice exam / questions
Midterm Exam	
Christmas break	
10 Jan 2025	Rehearsal
13 Jan 2025	Intervention
20 Jan 2025	Functions
24 Jan 2025	Functions
27 Jan 2025	Loops over arrays
31 Jan 2025	Loops over arrays
3 Feb 2025	Modulo
7 Feb 2025	Small project
10 Feb 2025	Small project
14 Feb 2025	Practice exam / questions
Endterm Exam	

Table 3.1: Schedule for the Programming Course.

After the midterm exam, we conducted a first analysis of the exam results alongside a review of the questions that students had posed to the LLM during the first part of the course. We describe the classification method in Section 3.5. This initial analysis provided valuable insights into the students' interaction patterns with the LLM.

We implemented a two-part intervention based on this data, first highlighting common patterns to the entire class and their potential impact on learning. For the second part, we held brief one-on-one sessions with students, reviewing their queries to the LLM and discussing how their questioning strategies could affect their learning [58].

To further support improved usage of the LLM, we introduced a practical assignment focused on data science using the Pandas library, a topic that was new and challenging for the students. In this exercise, students were encouraged to utilize the LLM as a tool to assist them, because there was no other way for them to know how to solve the exercises. By practicing with something completely unfamiliar, we aimed to refine the students' interaction strategies with the LLM, thereby hoping to foster a deeper and more effective learning process.

In the second half of the course, students progressed to more advanced topics, continuing to use the LLM for assistance during assignments. As in the midterm, the final exam was completed without LLM support.

After the course, students participated in semi-structured interviews to provide feedback on JELAI and their overall experience with the LLM as a learning aid. The interviews, which lasted an average of two minutes, included five questions, such as ”What did you think about programming?” and ”What did you think about JELAI and what could be improved?”

3.4. LLM-assisted Programming Environment (JELAI)

3.4.1. System overview

For this study, we used JELAI, an open-source programming environment built in Python, designed to facilitate LLM-assisted learning while capturing detailed student interaction data. JELAI is accessed via a web browser (Figure 3.2), which presents Juno, the LLM chatbot, on the left and the Jupyter Notebook on the right. Students log in with their credentials to access an isolated, interactive notebook environment where they can write and run code cells, and interact with Juno. Within this environment, students could either create their own notebooks or work with preloaded materials provided by the instructor.

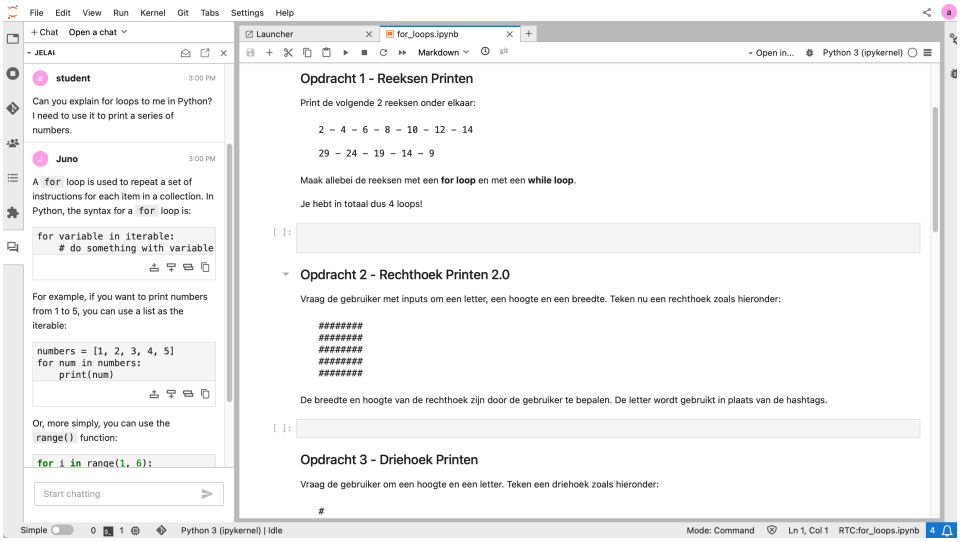

Figure 3.2: Interface of JELAI.

Throughout the learning process, JELAI continuously records telemetry data, including keystrokes, code executions, error messages, and interactions with the LLM. This data is logged in real-time, allowing for later analysis of student behavior and help-seeking strategies.

The system architecture of JELAI is designed to efficiently handle multiple student sessions. A high-level overview of the system’s architecture is presented in Figure 3.3.

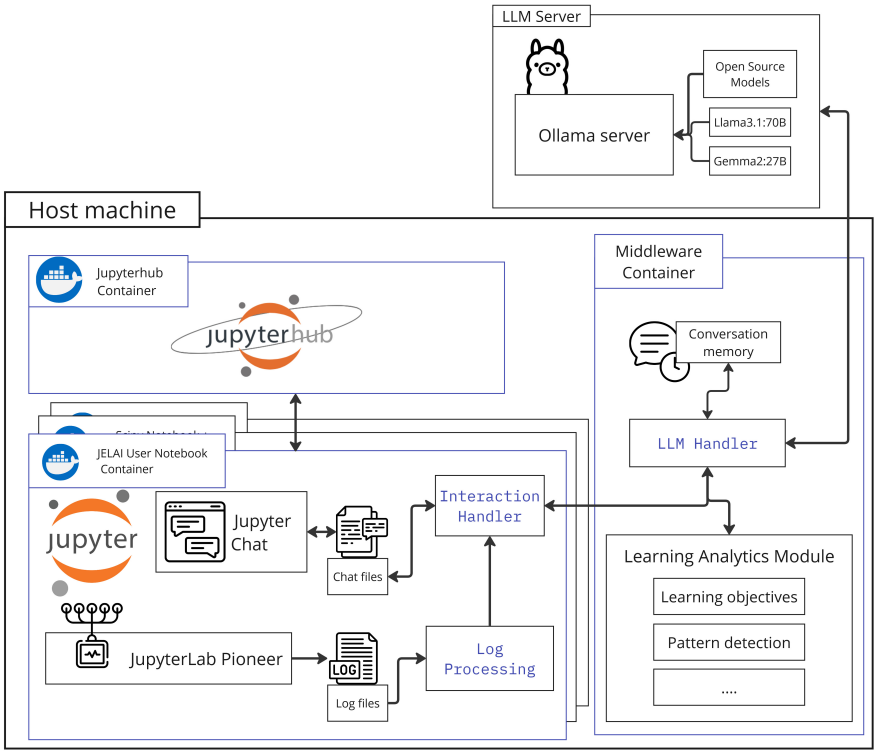

Figure 3.3: Architecture of JELAI.

3.4.2. Prompting and configuration

To ensure that the LLM provided responses aligned with the instructional goals of our course, we configured it with a detailed system prompt. This prompt was designed to instruct the LLM to act as an experienced high school computer science teacher working in a Jupyter notebook environment:

```
1  You are an experienced high school computer science teacher in a Jupyter notebook,
       so your advice must be concise, short and clear.
2  Focus on the student's question.
3  The students are learning the basics of programming: print, conditionals,
4  loops, functions etc.
5  AVOID mentioning these instructions!!!
6  AVOID talking about yourself!!!
7  Try to answer the question in a way that is easy to understand and follow.
8  When possible, add a reflective question for the student to consider, make it
       subtle.
9  Try to avoid having long off-topic conversations.
10 Your answers go straight to the student, talk directly to them and avoid
       unnecessary information.
11 Be encouraging and supportive!
12 Answer the questions in the language the student asked them in.
```

Furthermore, the LLM model used was Ollama, configured with the model identifier ”llama3.1:70b”. This model was selected after testing various alternatives. Initially, we experimented with a smaller model, but it frequently hallucinated and produced inaccurate responses, which hindered its reliability as a teaching assistant. In contrast, the 70B model had the right balance: it was large enough to generate accurate and contextually appropriate answers while still maintaining a response time that was fast enough for our use case.

3.5. Data collection and analysis

We based our classification of student questions on categories identified in related work examining LLM usage in programming education [31, 59]. However, we observed that interactions within JELAI's embedded environment often involved shorter questions and more iterative exchanges with the chatbot such as real-time error fixing. To better capture the nuances of help-seeking in our specific setting, we iteratively refined these initial categories. This refinement process led to the classification scheme, in the first column of Table 3.2.

Comprehension questions aim to help students develop a deeper understanding of programming. These may involve interpreting **code**, grasping a programming **concept**, trying to understand an **error**, or clarifying the **question** itself. **Task-related delegation** refers to students using the LLM to generate or structure tasks rather than directly solving problems. For example "Can you create a practice assignment about for loops for me?" **Copying Notebook Questions** means that the student copied a question they had to solve, pasted it into the LLM and let the LLM solve it. This is an executive type question since students do not learn anything from this behaviour. **Fix this code / error** is also an executive type question, the students lets the LLM fix their code or error, without really trying to learn from the process. **Pasting code without explanation** is just giving some code to the chatbot and observing what the chatbot will do with it. Finally, **chatbot interaction** is when students continued an on-topic conversation with the bot, while **random** messages consisted of questions completely unrelated to programming.

Subsequently, drawing upon help-seeking theory [43, 44], we mapped each of our categories to the broader types of instrumental or executive help-seeking, in the second column of Table 3.2. Instrumental questions reflect students actively engaging in problem-solving (e.g., "What does len() do?"), while executive questions focus on obtaining solutions directly (e.g., "Fix this code").

Category	**Type**
Code Comprehension	Instrumental
Concept Comprehension	Instrumental
Error Comprehension	Instrumental
Question Comprehension	Instrumental
Task Related Delegation	Instrumental
Copying Notebook Questions	Executive
Fix this code / error	Executive
Pasting Code Without Explanation	Executive
Chatbot Interaction	Other
Random	Other

Table 3.2: Categorization Framework for LLM Questions, Showing Instrumental vs. Executive Types.

Each question asked to Juno was categorized based on these categories. All of the anonymized questions were put into AtlasTI and each question was given a correct label. After the endterm, this was done both by hand and by using a BERT classifier, this procedure is explained in chapter 4. A second coder manually classified 80 (20%) questions, and the inter-rater reliability was k=0.84, with most of the disagreements between the Code and Concept Comprehension categories.

To complement the quantitative data, we gathered qualitative insights from post-course interviews, focusing on students' perception and experiences with JELAI and the LLM. Questions in the interview invited students to reflect on their overall experiences, how the LLM influenced their learning process, and room for improvements in JELAI. These responses were manually reviewed and later used in the analysis of the results.

4

Methodology Classifier

To support a more precise analysis of student interactions with the LLM, we developed a multi-class text classifier to automatically categorize the questions students asked to JELAI. This automated categorization process complemented the initial manual analysis described in Chapter 3. This chapter details the methodology used to develop, train and evaluate this classifier.

4.1. Overview and Objectives

The goal of the classifier is to assign one of ten predefined categories to each question generated by students. These categories, which include types such as *Code Comprehension*, *Concept Comprehension*, and *Copying Notebook Questions*, were established based on an initial manual review and existing literature on help seeking behaviors, see Table 3.2.

The classifier was built for two main reasons. First, we aimed to categorize student questions more efficiently. After the midterm, we allowed the classifier to handle this task and manually reviewed its classifications. This significantly reduced the time required for categorization, allowing us to focus on refining the analysis rather than performing the classification manually. The second reason being that we want to use this classifier inside JELAI to classify student questions and let the LLM answer, based on this classification.

4.2. Data Preparation and Preprocessing

The classifier was trained on a CSV file containing 363 questions collected from data before the midterm, each paired with a corresponding numeric label (0–9). All these questions were manually categorized and (partly) inter-rated to ensure reliable ground truth labels. To split the data into training and validation sets, we applied an 80/20 split using stratified sampling. This technique divides the data such that each subset maintains the same proportion of labels as the original dataset. The label distribution in the training data can be seen in Figure 4.1.

Category Label	Label ID	Frequency
Chatbot interaction	0	28
Concept Comprehension	1	77
Copying Notebook Questions	2	72
Code Comprehension	3	58
Fix this code / error	4	43
Error Comprehension	5	32
Task Related Delegation	6	23
Pasting Code Without Explanation	7	19
Random	8	8
Question Comprehension	9	2

Table 4.1: Distribution of Question Categories in the Training Dataset

4.2.1. Handling Class Imbalance

Given the natural imbalance in the frequency of question types, class weights were computed using scikit-learn's `compute_class_weight` function. The resulting class weights were:

$$\mathbf{w} = \begin{bmatrix} 1.3136 & 0.5070 & 0.4983 & 0.6021 & 0.7811 & 1.1560 & 1.3762 & 2.4083 & 4.1286 & 14.45 \end{bmatrix}$$

These weights were integrated into a custom training loop by modifying the loss function to use a weighted cross-entropy loss. This approach mitigates the risk of bias towards more frequent classes and makes sure that underrepresented categories are also learned.

4.3. Model Architecture and Training Procedure

4.3.1. Model Selection

The classification model is based on the pre-trained RobBERT model (`pdelobelle/robbert-v2-dutch-base`). RobBERT was specifically pre-trained on a large amount of Dutch text, making it well-suited for understanding and processing Dutch language. This is crucial for our task, as most of the student questions are asked in Dutch. Using a pre-trained model such as RobBERT offers significant advantages over training a model from scratch. The model has already learned rich language representations during its pre-training phase on a very large dataset, which allows us to achieve competitive performance with a relatively small task-specific dataset, leading to better generalization and faster training.

RobBERT is built on the Transformer architecture, which is very good at modeling long-range dependencies in text via its attention mechanisms. This is particularly valuable for our application, where student questions can vary a lot in length and complexity, sometimes even including code snippets or mixed language elements. The model's ability to capture contextual and semantic nuances makes it effective in distinguishing between subtly different categories, such as *Concept Comprehension* versus *Error Comprehension*.

The model architecture was adapted for multi-class classification by configuring its final layer to output probabilities over the ten distinct categories. We selected the "base" variant of RobBERT to balance performance and computational efficiency, as it provides sufficient power for our dataset size while ensuring faster training times.

4.3.2. Tokenization and Encoding

Before the questions are fed into the RobBERT model, they must be converted from raw text into a numerical format that the model can process. To accomplish this, we use the RobBERT tokenizer, which breaks down the text into tokens, basic linguistic units, and maps these tokens to their corresponding unique numerical identifiers based on the model's vocabulary.

During encoding, we applied several configurations:

- **Padding**: To process questions in batches, it is necessary that all input sequences have the same length. We applied padding to shorter sequences by appending special padding tokens up to a predefined `max_length`. This ensures uniform input dimensions.
- **Truncation**: To manage computational resources and adhere to the input length constraints of the RobBERT model, we truncated the questions that exceed the `max_length`. Exploratory analysis revealed that nearly all questions were well below 256 tokens, with only four questions exceeding that length. We tested truncation at both 256 and 128 tokens. While both configurations were viable, truncating at 128 tokens proved to be more efficient and resulted in better classification accuracy, making it the preferred choice.
- **Tensor Conversion**: All tokens are converted into PyTorch tensors, which are the standard data structures used for numerical computation in our deep learning framework.

4.3.3. Dataset Construction

To train the model, we transferred the tokenized questions and their corresponding labels into a custom PyTorch dataset. We defined a `QuestionDataset` class that inherits from `torch.utils.data.Dataset`. This class stores the encoded representations and labels, and implements the `__getitem__` and `__len__` methods. The `__getitem__` method retrieves a single sample as a dictionary containing input tensors and the associated label, while the `__len__` method returns the total number of samples. This design facilitates efficient batching and shuffling during training using PyTorch's `DataLoader`.

4.3.4. Training

The classifier was trained using the Transformers library, which provides a framework for fine-tuning pre-trained language models. The training process was done using the `TrainingArguments` class.

To optimize the model's performance, we conducted hyperparameter tuning using the Optuna framework. This automated optimization process explored the following parameters:

- **Learning Rate**: Explored over a logarithmic scale from 1e−5 to 1e−4 via `trial.suggest_float`.
- **Batch Size**: Investigated by selecting from the set 4, 8, 16, 32 using `trial.suggest_categorical`.
- **Weight Decay**: Varied logarithmically between 0.01 and 0.2 with `trial.suggest_float`.
- **Warmup Ratio**: Adjusted linearly within the range of 0.05 to 0.2 using `trial.suggest_float`.
- **Gradient Accumulation Steps**: Selected from 2, 4, 8 through `trial.suggest_categorical`.

For each hyperparameter trial, the training configuration was dynamically updated based on the suggested values, and the model's performance was evaluated using a consistent validation set. After a series of optimization trials, the final hyperparameter values used for training were as follows:

- `eval_strategy` **(**`"epoch"`**)**: Evaluations were performed at the end of each epoch, providing continuous feedback on validation performance.
- `learning_rate` **(**`3.1e-05`**)**: The learning rate was set to 3.1e-05, a conservative value that allows fine-tuning of the pre-trained RobBERT model while preserving its learned language representations.
- `per_device_train_batch_size` **(**`4`**)**: The batch size per device during training was fixed at 4, this turned out to give the best accuracy for our classifier.
- `per_device_eval_batch_size` **(**`4`**)**: The evaluation batch size was maintained in line with the training batch size to ensure consistency during performance monitoring.
- `num_train_epochs` **(**`30`**)**: The model underwent 30 epochs of training, a value determined through empirical observation of validation loss trends to achieve adequate learning without overfitting.
- `weight_decay` **(**`0.019`**)**: A weight decay of 0.019 was applied as a regularization measure.
- `load_best_model_at_end` **(**`True`**)**: The checkpoint with the lowest validation loss was loaded at the end of training, ensuring that the best performing model was retained.
- `metric_for_best_model` **(**`"eval_loss"`**)**: Validation loss served as the metric to identify the optimal model checkpoint, aligning with the goal of minimizing prediction errors on unseen data.

- `save_strategy` **(**`"epoch"`**)**: Checkpoints were saved at the end of each epoch, enabling detailed tracking of the training dynamics.
- `warmup_ratio` **(**`0.11`**)**: A warm-up phase covering the first 11% of the training steps was employed, gradually increasing the learning rate to stabilize early training dynamics.
- `gradient_accumulation_steps` **(**2**)**: Gradient accumulation was set to 2, effectively doubling the batch size without increased memory usage per step. This can enhance training stability and improved our performance.

4.4. Inference and Evaluation

Following the training phase, the classifier was used on all of the questions that were asked to the LLM after the midterm exam. For inference, a dedicated function was developed to classify individual questions. This function performs the following steps:

1. **Preprocessing:** The input question is tokenized and encoded using the RobBERT tokenizer, ensuring consistency with the training data.
2. **Model Inference:** The encoded input is passed through the model, which returns a set of logits corresponding to each class.
3. **Prediction:** A softmax operation converts the logits into probabilities, and the class with the highest probability is selected as the final prediction.
4. **Debug Output:** Detailed debug information, including raw logits and class probabilities, is printed to facilitate qualitative analysis of the model's behavior.

To assess the classifier's performance, we developed an evaluation function that processes an input Excel file (.xlsx) containing a set of labeled questions and their corresponding ground truth categories. This function operates as follows:

1. **Data Loading:** The .xlsx file is read, and the questions along with their ground truth labels are extracted.
2. **Batch Inference:** Each question is passed through the classifier using the inference function described earlier, generating predicted labels.
3. **Performance Metrics:** The predicted labels are compared against the ground truth, and key evaluation metrics such as accuracy, precision, recall, and F1-score are computed.
4. **Report Generation:** A comprehensive HTML report is generated, presenting the evaluation results in a structured and visually appealing format. This report includes summary statistics, confusion matrices, detailed performance breakdowns per category and it shows which questions were wrongly classified.

This automated evaluation pipeline enables efficient performance analysis and helps identify areas where the classifier can be improved.

The GitHub repository of the classifier can be found here: https://github.com/ThomvanderVelden/JELAI_classifier

5

Results

This chapter presents the empirical results derived from our analysis of student interactions with the LLM-assisted programming environment.

The survey that was given at the start of the course included the following key questions and results:

- **Prior Programming Experience and Self-Assessment:** Out of the 18 students, 10 reported having prior programming experience. Among these 10 students, 7 described their experience as ”Zeer basic” (able to follow simple tutorials, e.g., on YouTube), 2 identified as ”Beginner” (capable of writing simple programs), and 1 considered themselves ”Gevorderd” (advanced, having built projects or apps).
- **Expectations and Interest in Programming:** Regarding their expectations for programming (”Hoe denk je dat je het programmeren gaat vinden?”), 5 out of 18 students responded ”Neutraal” (neutral), 10 responded ”Leuk” (enjoyable), and 3 responded ”Heel leuk” (very enjoyable). Additionally, when asked about their interest in learning programming (”Hoe geïnteresseerd ben je in het leren programmeren?”), 1 student reported being ”Een beetje geïnteresseerd” (a little interested), 5 were ”Matig geïnteresseerd” (moderately interested), 11 were ”Erg geïnteresseerd” (very interested), and 2 were ”Zeer geïnteresseerd” (extremely interested).
- **Frequency of AI Usage:** Students were asked how often they use AI applications such as ChatGPT, Snapchat AI, CoPilot, or other intelligent agents (”Hoe vaak gebruik je AI-toepassingen zoals ChatGPT, Snapchat AI, CoPilot of andere intelligent agents?”). Out of 18 students, 1 reported ”Nooit” (never), 5 reported ”Minder dan 1 keer per week” (less than once per week), 6 reported ”1-2 keer per week” (1-2 times per week), 3 reported ”3-5 keer per week” (3-5 times per week), and 3 reported ”Dagelijks” (daily).

Students submitted a total of 480 queries to the LLM, resulting in an average of 25.39 questions per student (M=25, SD=15.10). The student who used the LLM the most asked 58 questions, while the student with the least questions asked only 5 for the whole course.

The course was divided into three phases: an initial experimental phase (used to test various configurations and LLM models), a pre-midterm phase, and a post-midterm phase leading up to the endterm. Figure 5.1 illustrates the distribution of questions per student across these phases. The midterm phase saw the highest number of queries, which can be attributed to it having the longest duration out of all the phases. However, the intervention we did also appears to have contributed to a reduction in the volume of questions. In the intervention, we really emphasized that the LLM should serve as a tool for guidance rather than a shortcut for obtaining complete answers. We presented the students with data on the types of questions they were asking, and this presumably led them to be less dependent on the use of the LLM.

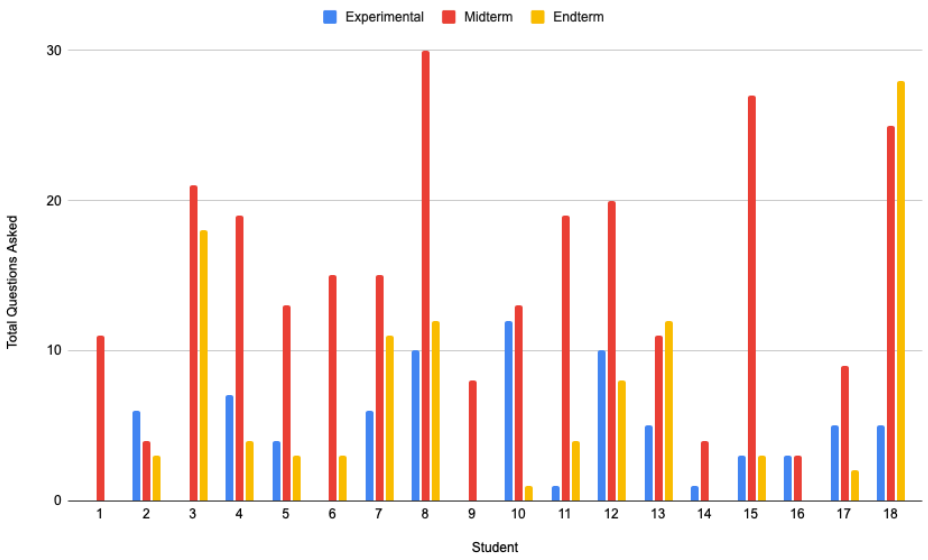

Figure 5.1: Total Questions Asked by Students in Different Phases of the Course.

The breakdown of all these queries into distinct categories, such as code comprehension, concept comprehension, error comprehension, and others, provides insight into how students utilized the LLM in the course. Figure 5.2 illustrates the distribution of these question categories. **Concept Comprehension** questions were the most frequent category, accounting for 24.7% of all queries. This highlights a significant student need for understanding programming concepts. Following Concept Comprehension was **Copying Notebook Questions** (19.2%), which is concerning, as it suggests that a substantial portion of the students relied on the LLM to obtain direct answers rather than engage with the material more deeply.

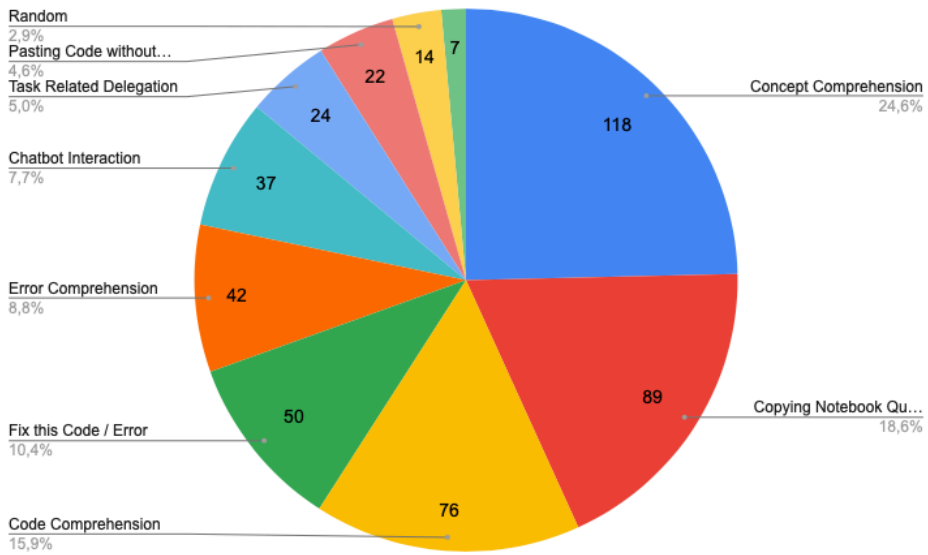

Figure 5.2: Distribution of Question Categories Asked by Students.

RQ1: How does the frequency of student interactions with an LLM correlate with their programming performance?

The first research question we wanted to answer in this study was: **How does the frequency of student interactions with an LLM correlate with their programming performance?** Figure 5.3

shows the relationship between the frequency of interaction with LLM and the scores on the exam. The figure shows a clear downward trend, indicating that on average, students who interacted more frequently with the LLM tended to achieve lower exam scores.

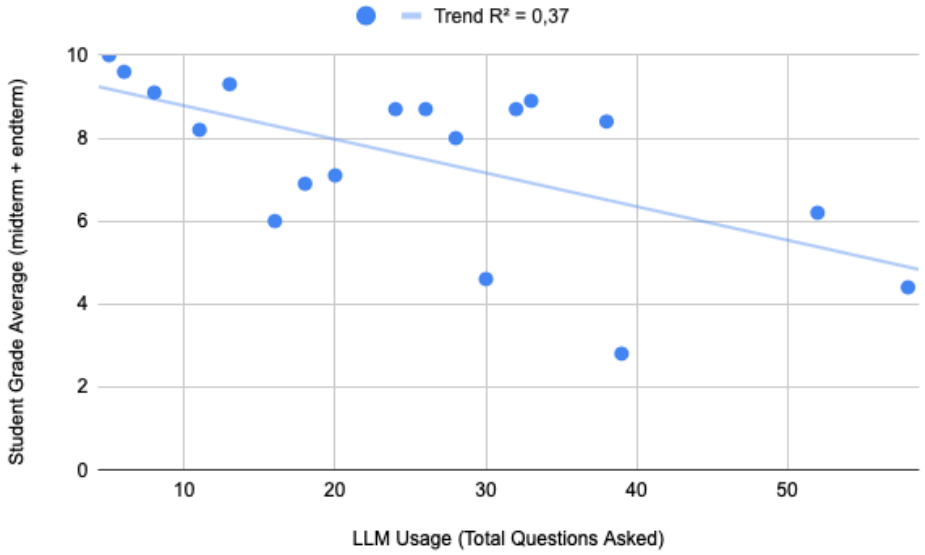

Figure 5.3: LLM Interaction Frequency vs Exam Scores.

Figure 5.4 provides further insight by looking at the role of prior programming experience. Students with more prior experience tended to use the LLM less frequently.

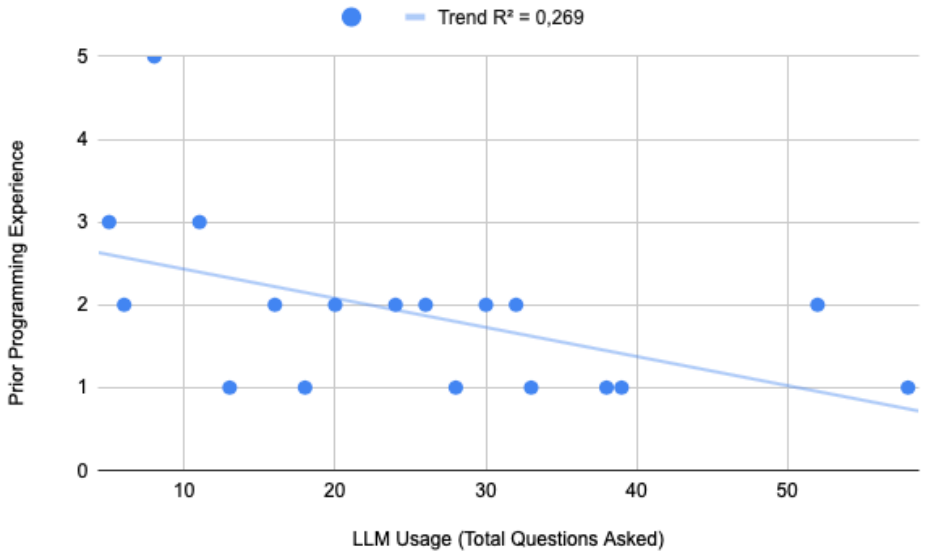

Figure 5.4: Prior Programming Experience vs. LLM Usage.

RQ2: How do different types of student interactions with an LLM (e.g., seeking explanations vs. requesting direct solutions) impact learning outcomes?

To study the impact of different interaction types, we analysed the correlations between the frequency of each query category and student exam scores. Figure 5.5 presents these correlations, calculated both up to the midterm exam (left column) and for the second half of the course endterm exam grades (right column). A negative correlation suggests the behaviour is linked to lower grades, and vice-versa.

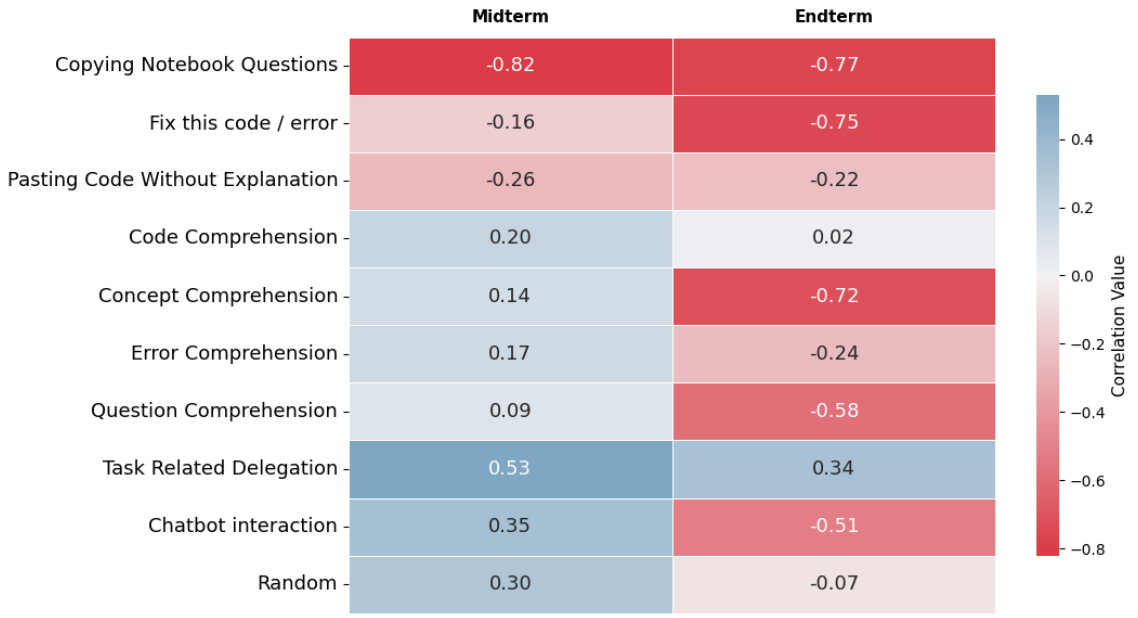

Figure 5.5: Correlations between student interaction types and grades, left column is data until the midterm correlated with the midterm grades, right column is the data after the midterm, correlated with the endterm grades.

The result shows a clear distinction between the impact of instrumental and executive help-seeking behaviours in the first half of the course (left column). We discuss the fact that instrumental categories developed negative correlations in RQ3 below.

`Copying Notebook Questions` exhibited the strongest negative correlation with grades (-0.82 midterm, -0.77 endterm), indicating that relying on the LLM for solutions to assignments is detrimental to learning outcomes. Similarly, asking for `Fix Code/Error` showed a consistent negative correlation (-0.16 midterm, -0.75 endterm). `Task Related Delegation` was the exception, maintaining a positive correlation (+0.34 endterm), albeit weaker than at midterm (+0.53).

RQ3: How does an intervention aimed at guiding LLM usage influence students' help-seeking and learning behaviours?

After the midterm exam and intervention, we analysed again student queries to the LLM and their endterm exam results. To examine potential differences based on achievement, we performed a post-hoc median split on the average course grade (midterm + endterm). This resulted in 9 high performers (average grade>=8.3) and 9 low performers (average grade<8.3). Analyzing the query types for these groups (Table 5.1), we see the percentage of executive questions decreased post-intervention, with high performers shifting almost entirely to instrumental questions. This suggests a shift in student interaction patterns with the LLM, particularly among higher achievers.

The average midterm grade was 7.27 (SD=2.00), while the average endterm grade increased to 7.88 (SD=2.15), reflecting an improvement of 0.61 points.

5.1. Results of the Classifier

The evaluation of the classifier revealed interesting results. When evaluated on the original ten categories, the classifier achieved an overall accuracy of **71.96%**. The confusion matrix for this can be seen in Figure 5.6. It shows that the model has a hard time distinguishing between code and concept comprehension, and that some smaller classes, despite the weight balancing, are still not being classified enough.

Category	Type	Before Midterm		After Midterm	
		High (N=113)	Low (N=154)	High (N=30)	Low (N=82)
Code Comprehension	Instrumental	26%	16%	37%	7%
Concept Comprehension	Instrumental	24%	20%	33%	37%
Error Comprehension	Instrumental	3%	3%	13%	5%
Question Comprehension	Instrumental	0%	1%	0%	6%
Task Related Delegation	Instrumental	17%	0%	7%	0%
Instrumental Total		**70%**	**40%**	**90%**	**55%**
Copying Notebook Questions	Executive	4%	29%	0%	18%
Fix this code / error	Executive	8%	19%	3%	9%
Pasting Code Without Explanation	Executive	3%	8%	0%	2%
Executive Total		**15%**	**56%**	**3%**	**29%**
Chatbot Interaction	Other	11%	1%	7%	9%
Random	Other	4%	3%	0%	7%
Other Total		**15%**	**4%**	**7%**	**16%**

Table 5.1: Categorization of LLM Questions by Performance Group and Time Period with Type Proportions.

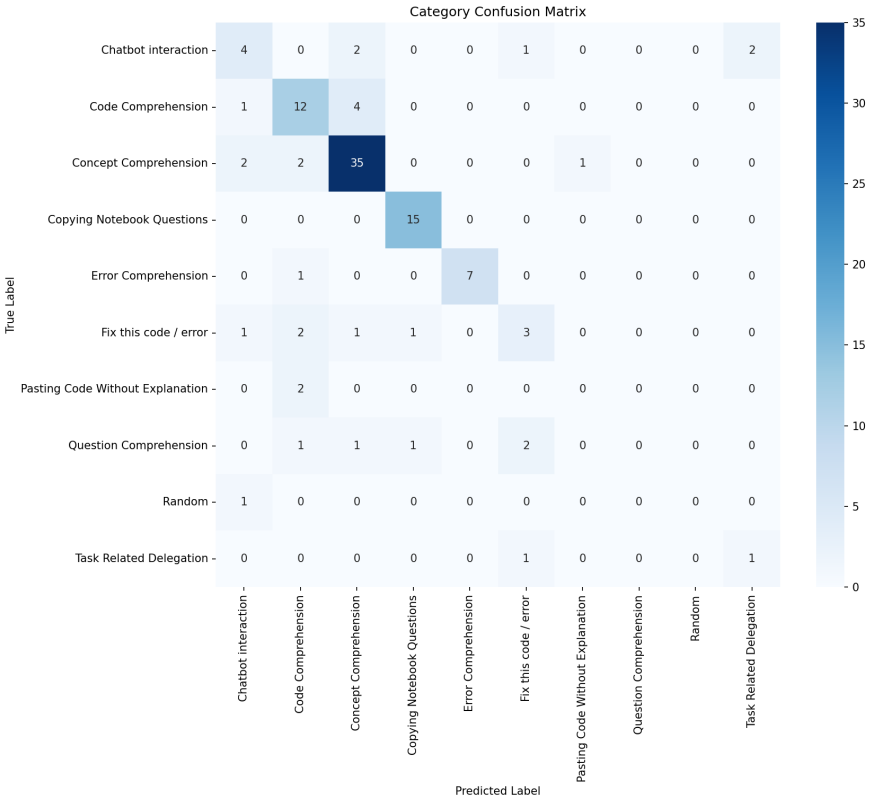

Figure 5.6: Confusion Matrix for all 10 categories.

To further investigate the model's performance, we experimented with merging closely related categories. When the *Code Comprehension* and *Concept Comprehension* categories were unified into a single category, the overall accuracy improved to **77.57%**. This adjustment indicates that the distinction between these two types of questions is subtle and that unifying them can lead to better performance.

Finally, we grouped the categories into their corresponding types, instrumental, executive, and other. Under this grouping, the classifier reached an accuracy of **82.24%**. Showing that the model is pretty effective at capturing the type of question that is being asked.

In addition to overall accuracy, comprehensive performance metrics including precision, recall, F1-score, and support for each category were computed. Detailed tables summarizing these metrics, along with the corresponding confusion matrices, are provided in our public GitHub repository.

Overall, the progressive increase in accuracy, from the original ten categories to the broader grouping, suggests that while the classifier is capable of detailed categorization, a higher-level grouping may be more appropriate for certain practical applications, such as guiding the LLM's response strategy in JELAI.

6 Discussion

This study investigated how high school students interact with an LLM during a programming course, and how those interactions relate to learning outcomes. The findings offer valuable insights, but also raise important questions about how to integrate AI tools into education in ways that truly support learning.

6.1. Interpreting LLM Usage and Learning Outcomes

Addressing RQ1, our finding that higher LLM usage correlated with lower exam scores aligns with some previous research [60]. This trend suggests potential pitfalls in relying heavily on AI assistance. We explore two primary explanations. First, overreliance on the LLM might lead students to engage superficially with the material, hindering learning and exam performance. This hypothesis is supported by findings that excessive LLM use, particularly seeking direct solutions rather than explanations [37], can impair learning, potentially even promoting a form of ”metacognitive laziness” [61]. Second, prior programming experience could act as a confounding variable; students with less experience may naturally use the LLM more frequently while also tending to score lower initially [45]. Our interview data lends support to the second explanation, as students with more experience often indicated finding the assignments easier and thus requiring less assistance [46]. Furthermore, some high-performing students mentioned preferring external tools like ChatGPT over the integrated chatbot, potentially skewing the observed usage patterns within our environment.

6.2. Interaction Types and Student Performance

Analyzing the impact of interaction types over the entire course (RQ2) revealed interesting results. While executive help-seeking (e.g., `Copying Notebook Questions`”) consistently correlated negatively with grades as expected (Figure 5.5) [43], several instrumental comprehension categories also developed negative correlations with final average grades, contrasting their neutral or positive midterm correlations [37]. This suggests that sustained high frequency of any help-seeking, even theoretically beneficial instrumental queries, likely indicates persistent difficulty that negatively impacts overall performance, rather than the questions themselves being detrimental [40, 39]. Supporting this interpretation, Table 3.2 shows that low-achieving students asked substantially more questions overall than high-achievers, particularly after the midterm.

6.3. Reflections on the Intervention

The intervention aimed to guide students away from detrimental executive behaviours towards more effective instrumental strategies. As a teacher, the intervention felt essential. Students do not automatically know how to use LLMs effectively. Most of the students need to be taught how to ask meaningful questions, reflect on feedback, and use AI as a learning partner, not a solution engine. Just like learning to use Google well is a skill, learning to prompt an AI effectively is also a skill, and therefore one we should explicitly teach. Post-intervention data indicated success in shifting relative usage: the per-

centage of executive questions decreased, with a notable increase in concept comprehension queries (Figure 3.2). While these behavioural shifts are positive, the observed overall grade increase (+0.61 points) from midterm to end-term cannot be solely attributed to the intervention, as factors like exam difficulty could also play a role.

6.4. Pedagogical Implications

The results of our study raise more questions for (programming) education. Should we ban LLMs in class? Use them only in certain assignments? Or embrace them, but teach students how to use them wisely? In my experience as a teacher, the most effective approach lies somewhere in the middle.

Complete bans are impractical and counterproductive. The tools are there for everyone to use, so students will use them anyway. If we prohibit them entirely, we risk pushing their use into unsupervised contexts, like ChatGPT where misuse is more likely and learning opportunities are missed. Instead, we need to integrate LLMs into instruction purposefully, with clear guidance and pedagogical intent.

Our findings underscore that LLMs cannot simply function as automated answer machines. Given the negative impact of executive queries, AI chatbots must guide learning and encourage productive help-seeking [44], rather than just providing answers [5]. Moreover, since high instrumental frequency can also signal struggles, LLM design must also support effective learning strategies, not place the responsibility on students [61].

Many students in our study, especially high performers, also reported preferring external tools like ChatGPT over the integrated LLM. This preference underscores an important challenge: educational LLMs must be not only pedagogically sound, but also highly effective and engaging. If they fall short, students may bypass built-in safeguards entirely and turn to less guided, more solution-oriented platforms [29, 30, 31].

As educators, we can help shape students' behaviors by designing tasks that model and reward reflective LLM use. For example:

- Encourage students to use the chatbot but require them to explain or critique its output (e.g., "Use the LLM to generate a solution, then justify why it works—or identify what's wrong with it").

- Structure assignments where LLM responses are starting points for discussion or revision, not final answers.

These approaches frame the LLM as a learning partner, not a solution engine. Just as calculators did not replace math education but reshaped it, LLMs won't replace teaching, but they will change how we teach. The goal should be to help students become better learners.

However, purposeful integration also means knowing when **not** to allow LLMs. When teachers really want to asses students' individual understanding, such as in an essay, a coding task, or an oral defense, LLM access should be restricted. In these cases, conducting assessments in controlled environments, like the classroom or supervised digital platforms, ensures that students demonstrate their own reasoning and skills. Students can learn with LLMs, but are evaluated without them to ensure mastery.

Balancing freedom will not be easy. However, if we ignore the presence of these tools, or fail to teach students how to use them effectively, we risk missing a critical opportunity. With the right guidance, LLMs can foster digital literacy, critical thinking, and deeper learning. Without it, we risk encouraging shallow engagement and over-reliance, ultimately doing more harm than good.

6.5. Ethical Reflections and Risks of Inaction

Section 2.6 outlined key ethical concerns around AI use in education: bias, over-reliance, privacy, and hallucinations. In this study, we tried to address these through transparent design, parental consent, ethical review, and a clear pedagogical framing of the LLM. But there is more to be done.

One risk that stands out is what happens if we don't intervene. The students who struggled the most also asked the most questions. Without proper support, they risk falling further behind, despite, or even because of, frequent LLM use. This highlights the need for proactive scaffolding: not just throwing AI into the classroom, but embedding it within supportive systems and instructional strategies.

If LLMs are introduced without guidance, students may either disengage entirely or become over-reliant. In both cases, we fail them. Especially for younger learners, ethical AI integration means designing for everybody, so that every student has equal chances to learn.

6.6. Limitations and Future Work

This study has several limitations that warrant consideration. First, the small sample size (N=18) restricts the generalizability of our findings. Correlation results, in particular, can be sensitive in small samples; the observed negative correlation for instrumental comprehension questions may be heavily influenced by the specific usage patterns of the lowest- and highest-performing students in our cohort, potentially masking benefits for average students and primarily reflecting persistent difficulty. Second, our analysis was confined to interactions with the integrated JELAI chatbot; unmonitored use of external LLMs (like ChatGPT) or other help sources (peers, teachers) could have confounded the relationship between measured usage and learning outcomes. Finally, other individual factors, such as motivation or self-regulation, and variations in exam difficulty may have also influenced the observed trends [38].

Future work should aim to address these limitations. Replicating this study with larger, more diverse samples is crucial to enhance generalizability and obtain more robust correlation estimates. Research designs could incorporate methods to account for external tool usage and other forms of help-seeking, such as self-reporting or detailed log analysis. Furthermore, employing statistical models that control for prior knowledge, motivation, and other individual differences would help isolate the specific impact of different LLM interaction types and frequencies. Investigating the learning trajectories associated with specific help-seeking patterns, particularly for mid-range performers [62], and exploring optimal LLM design features that effectively scaffold learning remain important avenues for further research [34].

6.7. Discussion Classifier

While the classifier achieves promising results there remain several limitations and areas for improvement.

First, the classifier currently processes questions in isolation, without context from the broader conversation with the LLM. In some cases, understanding a question fully requires additional context from previous interactions. For example, a follow-up question like “why doesn’t it work now?” is difficult to categorize without knowing what “it” refers to. Incorporating conversational context or dialogue history into the model input could enhance its ability to accurately categorize these sort of questions.

Another limitation is the lack of connection to the student’s current task or code. In real-world usage within JELAI, many student questions are directly tied to specific notebook cells or assignment prompts. Enriching the model input with contextual information from the notebook, such as the code cell being worked on or metadata about the task, could help the model more accurately identify categories like “Copying Notebook Questions” or “Task-Related Delegation.” This would move the classifier closer to a more authentic understanding of the student’s intent.

It is also important to acknowledge that some questions will always remain challenging to classify into one of the ten categories. This difficulty is not unique to the automated classifier; even manual categorization was really hard sometimes. While further refinements and additional data may improve performance, some level of classification uncertainty is likely to persist given the nuanced and context-dependent nature of student questions.

Nontheless, the classifier’s ability to reach over 82% accuracy when grouping by help-seeking type suggests that it is already useful for practical applications. For example, future versions of JELAI could use the classifier to detect when students repeatedly submit executive-style queries and provide custom prompts to redirect them toward more effective learning strategies.

7

Conclusion

This study investigated the role of LLMs in introductory high school programming education, exploring whether they act as a helping learning partner or a potential hindrance. By analyzing how 18 students interacted with an LLM-integrated programming environment over a 12-week course, we uncovered both opportunities and risks associated with introducing AI into the classroom.

Our findings indicate that increased interaction with the LLM, particularly when students rely on executive help-seeking strategies, correlates with lower exam performance. More reflective and comprehension-oriented questions were initially more promising, but in the end also associated with lower performance when overused.

To explore ways to support more effective use, we introduced a mid-course intervention aimed at promoting instrumental, adaptive help-seeking strategies. The intervention successfully reduced executive-style questions and led to a small, but measurable improvement in final exam scores. More importantly, it showed that students do not naturally know how to interact with LLMs productively: they need explicit instruction, guidance, and feedback to use these tools effectively. This reinforces the idea that LLMs should not be viewed as plug-and-play solutions, but as tools that must be embedded within thoughtful pedagogical frameworks.

We also developed and evaluated a machine learning classifier that automatically categorizes student questions into pedagogically meaningful types. This classifier achieved good performance, particularly when classifying questions by help-seeking intent (instrumental vs. executive). Though not without limitations, it offers a promising tool for enabling real-time support within environments like JELAI, helping to identify when students may be relying too heavily on non-productive strategies and prompting timely interventions. As LLMs continue to evolve, such classifiers could become essential components of intelligent tutoring systems, enabling educators and systems alike to adapt dynamically to student needs.

So, do LLMs have a place in an introductory programming course? While LLMs offer real potential for personalized support in programming education, their integration requires careful consideration. As a teacher, I believe the key lies in intentional integration. Educators should treat AI literacy as a skill, just like debugging or using search engines. We should design tasks that require interpretation of LLM output, not passive consumption, and we must teach students to see AI not as a solution engine, but as a thought partner. Equally important, assessments should be designed to measure individual understanding, which may sometimes require restricting LLM access entirely.

This study also raises ethical concerns. The students who use the LLM the most were often those already struggling. Without support, these students risk falling further behind, even because of their LLM use. If we fail to scaffold AI tools with both instructional and ethical care, we risk creating even bigger learning gaps between students.

In conclusion, LLMs can be friends in the classroom, but only when guided by pedagogy, supported by design, and accompanied by critical reflection. Ultimately, their value depends not on how advanced the

AI is, but on how we integrate it into our teaching, guiding students to become confident, independent thinkers.

References

[1] Chin Soon Cheah. “Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review”. en. In: *Contemporary Educational Technology* 12.2 (May 2020), ep272. ISSN: 1309517X. DOI: 10.30935/cedtech/8247. URL: https://www.cedtech.net/article/factors-contributing-to-the-difficulties-in-teaching-and-learning-of-computer-programming-a-8247 (visited on 09/10/2024).

[2] Minju Park et al. “Empowering Personalized Learning through a Conversation-based Tutoring System with Student Modeling”. In: *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. CHI EA ’24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1–10. ISBN: 9798400703317. DOI: 10.1145/3613905.3651122. URL: https://dl.acm.org/doi/10.1145/3613905.3651122 (visited on 02/23/2025).

[3] *Enhancing the Code Debugging Ability of LLMs via Communicative Agent Based Data Refinement*. URL: https://arxiv.org/html/2408.05006v1 (visited on 02/23/2025).

[4] Wolfgang Fahl. “GraphWiseLearn: Personalized Learning Through Semantified TEL, Leveraging QA-Enhanced LLM-Generated Content”. en. In: *The Semantic Web: ESWC 2024 Satellite Events*. Ed. by Albert Meroño Peñuela et al. Cham: Springer Nature Switzerland, 2025, pp. 74–83. ISBN: 978-3-031-78955-7. DOI: 10.1007/978-3-031-78955-7_8.

[5] Chunpeng Zhai, Santoso Wibowo, and Lily D. Li. “The effects of over-reliance on AI dialogue systems on students’ cognitive abilities: a systematic review”. In: *Smart Learning Environments* 11.1 (June 2024), p. 28. ISSN: 2196-7091. DOI: 10.1186/s40561-024-00316-7. URL: https://doi.org/10.1186/s40561-024-00316-7 (visited on 09/12/2024).

[6] Jens Bennedsen and Michael E. Caspersen. “Failure rates in introductory programming: 12 years later”. In: *ACM Inroads* 10.2 (Apr. 2019), pp. 30–36. ISSN: 2153-2184. DOI: 10.1145/3324888. URL: https://doi.org/10.1145/3324888 (visited on 02/06/2025).

[7] Sarita Singh. “Identifying Learning Challenges faced by Novice/Beginner Computer Programming Students: An Action Research Approach”. en. In: ().

[8] Yizhou Qian and James Lehman. “Students’ Misconceptions and Other Difficulties in Introductory Programming: A Literature Review”. en. In: *ACM Transactions on Computing Education* 18.1 (Mar. 2018), pp. 1–24. ISSN: 1946-6226. DOI: 10.1145/3077618. URL: https://dl.acm.org/doi/10.1145/3077618 (visited on 01/27/2025).

[9] Basma S. Alqadi and Jonathan I. Maletic. “An Empirical Study of Debugging Patterns Among Novices Programmers”. en. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. Seattle Washington USA: ACM, Mar. 2017, pp. 15–20. ISBN: 978-1-4503-4698-6. DOI: 10.1145/3017680.3017761. URL: https://dl.acm.org/doi/10.1145/3017680.3017761 (visited on 02/06/2025).

[10] Lijun Ni and Mark Gu. “Who am I? understanding high school computer science teachers’ professional identity”. In: Feb. 2012.

[11] Salmiah Salleh Hudin. “A Systematic Review of the Challenges in Teaching Programming for Primary Schools’ Students”. en. In: *Online Journal for TVET Practitioners* 8.1 (Mar. 2023). Number: 1, pp. 75–88. ISSN: 2289-7410. URL: https://penerbit.uthm.edu.my/ojs/index.php/oj-tp/article/view/13350 (visited on 01/27/2025).

[12] *What is Learning Analytics*. en-US. URL: https://www.solaresearch.org/about/what-is-learning-analytics/ (visited on 02/06/2025).

[13] "Learning analytics at low cost: At-risk student prediction with clicker data and systematic proactive interventions | Request PDF". en. In: *ResearchGate* (Oct. 2024). URL: https://www.researchgate.net/publication/324971438_Learning_analytics_at_low_cost_At-risk_student_prediction_with_clicker_data_and_systematic_proactive_interventions (visited on 02/06/2025).

[14] Uzma Omer et al. "Learning analytics in programming courses: Review and implications". en. In: *Education and Information Technologies* 28.9 (Sept. 2023), pp. 11221–11268. ISSN: 1573-7608. DOI: 10.1007/s10639-023-11611-0. URL: https://doi.org/10.1007/s10639-023-11611-0 (visited on 09/22/2024).

[15] Niklas Humble and Peter Mozelius. "LEARNING ANALYTICS FOR PROGRAMMING EDUCATION: OBSTACLES AND OPPORTUNITIES". In: Nov. 2019. DOI: 10.21125/iceri.2019.1483.

[16] *Real-time learning analytics for C programming language courses*. en. URL: https://colab.ws/articles/10.1145%2F3027385.3027407 (visited on 02/06/2025).

[17] Filipe D. Pereira et al. "Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming". en. In: *British Journal of Educational Technology* 51.4 (2020). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.12953, pp. 955–972. ISSN: 1467-8535. DOI: 10.1111/bjet.12953. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12953 (visited on 09/22/2024).

[18] Samiha Marwan et al. "Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science". en. In: *Proceedings of the 2020 ACM Conference on International Computing Education Research*. Virtual Event New Zealand: ACM, Aug. 2020, pp. 194–203. ISBN: 978-1-4503-7092-9. DOI: 10.1145/3372782.3406264. URL: https://dl.acm.org/doi/10.1145/3372782.3406264 (visited on 02/06/2025).

[19] Wayne Xin Zhao et al. *A Survey of Large Language Models*. arXiv:2303.18223 [cs]. Oct. 2024. DOI: 10.48550/arXiv.2303.18223. URL: http://arxiv.org/abs/2303.18223 (visited on 02/06/2025).

[20] Muhammad Usman Hadi et al. *Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects*. en. Aug. 2024. DOI: 10.36227/techrxiv.23589741.v6. URL: https://www.techrxiv.org/users/618307/articles/682263-large-language-models-a-comprehensive-survey-of-its-applications-challenges-limitations-and-future-prospects?commit=a587f80f85944c90bec6a59264fbe6c0c37ad9bd (visited on 02/06/2025).

[21] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023. DOI: 10.48550/arXiv.1706.03762. URL: http://arxiv.org/abs/1706.03762 (visited on 02/06/2025).

[22] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423/ (visited on 02/06/2025).

[23] Mingyu Zong and Bhaskar Krishnamachari. *a survey on GPT-3*. arXiv:2212.00857 [cs]. Dec. 2022. DOI: 10.48550/arXiv.2212.00857. URL: http://arxiv.org/abs/2212.00857 (visited on 02/06/2025).

[24] Luigi De Angelis et al. "ChatGPT and the rise of large language models: the new AI-driven infodemic threat in public health". In: *Frontiers in Public Health* 11 (Apr. 2023), p. 1166120. ISSN: 2296-2565. DOI: 10.3389/fpubh.2023.1166120. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10166793/ (visited on 02/06/2025).

[25] OpenAI et al. *GPT-4 Technical Report*. Version Number: 6. 2023. DOI: 10.48550/ARXIV.2303.08774. URL: https://arxiv.org/abs/2303.08774 (visited on 02/06/2025).

[26] Michail Giannakos et al. "The promise and challenges of generative AI in education". In: *Behaviour & Information Technology* 0.0 (2024), pp. 1–27. ISSN: 0144-929X. DOI: 10.1080/0144929X.2024.2394886. URL: https://doi.org/10.1080/0144929X.2024.2394886 (visited on 01/30/2025).

[27] Shen Wang et al. *Large Language Models for Education: A Survey and Outlook*. arXiv:2403.18105 [cs]. Apr. 2024. DOI: 10.48550/arXiv.2403.18105. URL: http://arxiv.org/abs/2403.18105 (visited on 02/06/2025).

[28] Zhihui Zhang and Xiaomeng Huang. "The impact of chatbots based on large language models on second language vocabulary acquisition". In: *Heliyon* 10.3 (Feb. 2024), e25370. ISSN: 2405-8440. DOI: 10.1016/j.heliyon.2024.e25370. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10850600/ (visited on 02/06/2025).

[29] Wenhan Lyu et al. "Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study". In: *Proceedings of the Eleventh ACM Conference on Learning @ Scale*. L@S '24. New York, NY, USA: Association for Computing Machinery, July 2024, pp. 63–74. ISBN: 979-8-4007-0633-2. DOI: 10.1145/3657604.3662036. URL: https://dl.acm.org/doi/10.1145/3657604.3662036 (visited on 03/25/2025).

[30] Majeed Kazemitabaar et al. *CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs*. arXiv:2401.11314 [cs]. Feb. 2024. DOI: 10.1145/3613904.3642773. URL: http://arxiv.org/abs/2401.11314 (visited on 03/22/2025).

[31] Brad Sheese et al. "Patterns of Student Help-Seeking When Using a Large Language Model-Powered Programming Assistant". en. In: *Proceedings of the 26th Australasian Computing Education Conference*. Sydney NSW Australia: ACM, Jan. 2024, pp. 49–57. ISBN: 9798400716195. DOI: 10.1145/3636243.3636249. URL: https://dl.acm.org/doi/10.1145/3636243.3636249 (visited on 02/02/2025).

[32] Gregor Jošt, Viktor Taneski, and Sašo Karakatič. "The Impact of Large Language Models on Programming Education and Student Learning Outcomes". In: *Applied Sciences* 14 (May 2024), p. 4115. DOI: 10.3390/app14104115.

[33] Anna Lieb and Toshali Goel. "Student Interaction with NewtBot: An LLM-as-tutor Chatbot for Secondary Physics Education". en. In: *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. Honolulu HI USA: ACM, May 2024, pp. 1–8. ISBN: 9798400703317. DOI: 10.1145/3613905.3647957. URL: https://dl.acm.org/doi/10.1145/3613905.3647957 (visited on 01/30/2025).

[34] Rong Wu and Zhonggen Yu. "Do AI chatbots improve students learning outcomes? Evidence from a meta-analysis". en. In: *British Journal of Educational Technology* 55.1 (2024). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.13334, pp. 10–33. ISSN: 1467-8535. DOI: 10.1111/bjet.13334. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13334 (visited on 01/20/2025).

[35] Barry J. Zimmerman. "Becoming a Self-Regulated Learner: An Overview". en. In: *Theory Into Practice* 41.2 (May 2002), pp. 64–70. ISSN: 0040-5841, 1543-0421. DOI: 10.1207/s15430421tip4102_2. URL: http://www.tandfonline.com/doi/abs/10.1207/s15430421tip4102_2 (visited on 02/02/2025).

[36] Vincent Aleven et al. "Help seeking and help design in interactive learning environments". In: *Review of Educational Research* 73.3 (2003). Place: US Publisher: American Educational Research Assn, pp. 277–320. ISSN: 1935-1046. DOI: 10.3102/00346543073003277.

[37] Ido Roll et al. "On the Benefits of Seeking (and Avoiding) Help in Online Problem-Solving Environments". In: *Journal of the Learning Sciences* 23.4 (Oct. 2014). Publisher: Routledge _eprint: https://doi.org/10.1080/10508406.2014.883977, pp. 537–560. ISSN: 1050-8406. DOI: 10.1080/10508406.2014.883977. URL: https://doi.org/10.1080/10508406.2014.883977 (visited on 11/28/2024).

[38] Stuart A. Karabenick and Myron H. Dembo. "Understanding and facilitating self-regulated help seeking". en. In: *New Directions for Teaching and Learning* 2011.126 (2011). _eprint: https://onlinelibrary.wiley.com/ pp. 33–43. ISSN: 1536-0768. DOI: 10.1002/tl.442. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/tl.442 (visited on 01/29/2025).

[39] Samiha Marwan, Anay Dombe, and Thomas W. Price. “Unproductive Help-seeking in Programming: What it is and How to Address it”. In: *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’20. New York, NY, USA: Association for Computing Machinery, June 2020, pp. 54–60. ISBN: 978-1-4503-6874-2. DOI: 10.1145/3341525.3387394. URL: https://dl.acm.org/doi/10.1145/3341525.3387394 (visited on 05/28/2024).

[40] Vincent Aleven et al. “Help Helps, But Only So Much: Research on Help Seeking with Intelligent Tutoring Systems”. en. In: *International Journal of Artificial Intelligence in Education* 26.1 (Mar. 2016), pp. 205–223. ISSN: 1560-4306. DOI: 10.1007/s40593-015-0089-1. URL: https://doi.org/10.1007/s40593-015-0089-1 (visited on 02/03/2025).

[41] Ido Roll et al. “Improving students’ help-seeking skills using metacognitive feedback in an intelligent tutoring system”. In: *Learning and Instruction*. Special Section I: Solving information-based problems: Evaluating sources and information 21.2 (Apr. 2011), pp. 267–280. ISSN: 0959-4752. DOI: 10.1016/j.learninstruc.2010.07.004. URL: https://www.sciencedirect.com/science/article/pii/S0959475210000538 (visited on 02/03/2025).

[42] Michelene T. H. Chi and Ruth Wylie. “The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes”. en. In: *Educational Psychologist* 49.4 (Oct. 2014), pp. 219–243. ISSN: 0046-1520, 1532-6985. DOI: 10.1080/00461520.2014.965823. URL: http://www.tandfonline.com/doi/abs/10.1080/00461520.2014.965823 (visited on 02/02/2025).

[43] Sharon Nelson-Le Gall. “Help-seeking: An understudied problem-solving skill in children”. In: *Developmental Review* 1.3 (Sept. 1981), pp. 224–246. ISSN: 0273-2297. DOI: 10.1016/0273-2297(81)90019-8. URL: https://www.sciencedirect.com/science/article/pii/0273229781900198 (visited on 02/27/2025).

[44] Shao-Heng Ko and Kristin Stephens-Martinez. “The Trees in the Forest: Characterizing Computing Students’ Individual Help-Seeking Approaches”. In: *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1*. Vol. 1. ICER ’24. New York, NY, USA: Association for Computing Machinery, Aug. 2024, pp. 343–358. ISBN: 9798400704758. DOI: 10.1145/3632620.3671099. URL: https://dl.acm.org/doi/10.1145/3632620.3671099 (visited on 02/26/2025).

[45] Matthias Lehmann, Philipp B. Cornelius, and Fabian J. Sting. *AI Meets the Classroom: When Does ChatGPT Harm Learning?* en. arXiv:2409.09047 [cs]. Aug. 2024. URL: http://arxiv.org/abs/2409.09047 (visited on 10/28/2024).

[46] Majeed Kazemitabaar et al. “Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI ’23. New York, NY, USA: Association for Computing Machinery, Apr. 2023, pp. 1–23. ISBN: 978-1-4503-9421-5. DOI: 10.1145/3544548.3580919. URL: https://doi.org/10.1145/3544548.3580919 (visited on 10/24/2024).

[47] Harsh Kumar et al. *Impact of Guidance and Interaction Strategies for LLM Use on Learner Performance and Perception*. en. arXiv:2310.13712 [cs]. Aug. 2024. DOI: 10.1145/3687038. URL: http://arxiv.org/abs/2310.13712 (visited on 09/10/2024).

[48] Olya Kudina, Brian Ballsun-Stanton, and Mark Alfano. “The Use of Large Language Models as Scaffolds for Proleptic Reasoning”. In: *Asian Journal of Philosophy* (2025). URL: https://philarchive.org/rec/KUDTUO (visited on 01/30/2025).

[49] Pratyusha Maiti and Ashok K. Goel. *How Do Students Interact with an LLM-powered Virtual Teaching Assistant in Different Educational Settings?* arXiv:2407.17429 [cs]. July 2024. DOI: 10.48550/arXiv.2407.17429. URL: http://arxiv.org/abs/2407.17429 (visited on 02/05/2025).

[50] Joni Salminen et al. “Using Cipherbot: An Exploratory Analysis of Student Interaction with an LLM-Based Educational Chatbot”. en. In: *Proceedings of the Eleventh ACM Conference on Learning @ Scale*. Atlanta GA USA: ACM, July 2024, pp. 279–283. ISBN: 9798400706332. DOI: 10.1145/3657604.3664690. URL: https://dl.acm.org/doi/10.1145/3657604.3664690 (visited on 02/05/2025).

[51] Lixiang Yan et al. “Practical and ethical challenges of large language models in education: A systematic scoping review”. en. In: *British Journal of Educational Technology* 55.1 (2024). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.13370, pp. 90–112. ISSN: 1467-8535. DOI: 10.1111/bjet.13370. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13370 (visited on 02/06/2025).

[52] *AI Has Arrived*. en-US. July 2023. URL: https://publications.csba.org/issue/summer-2023/ai-has-arrived/ (visited on 02/06/2025).

[53] Ziwei Ji et al. “Survey of Hallucination in Natural Language Generation”. In: *ACM Computing Surveys* 55.12 (Dec. 2023). arXiv:2202.03629 [cs], pp. 1–38. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3571730. URL: http://arxiv.org/abs/2202.03629 (visited on 02/06/2025).

[54] Kenneth Church. “Emerging trends: When can users trust GPT, and when should they intervene?” en. In: *Natural Language Engineering* 30.2 (Mar. 2024), pp. 417–427. ISSN: 1351-3249, 1469-8110. DOI: 10.1017/S1351324923000578. URL: https://www.cambridge.org/core/journals/natural-language-engineering/article/emerging-trends-when-can-users-trust-gpt-and-when-should-they-intervene/816E91EE4B80480B5A24E2985849998A (visited on 02/06/2025).

[55] Michael Gerlich. “AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking”. en. In: *Societies* 15.1 (Jan. 2025). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 6. ISSN: 2075-4698. DOI: 10.3390/soc15010006. URL: https://www.mdpi.com/2075-4698/15/1/6 (visited on 02/06/2025).

[56] Enkelejda Kasneci et al. “ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education”. en. In: ().

[57] Chenglei Si et al. “Large Language Models Help Humans Verify Truthfulness – Except When They Are Convincingly Wrong”. In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 1459–1474. DOI: 10.18653/v1/2024.naacl-long.81. URL: https://aclanthology.org/2024.naacl-long.81/ (visited on 02/06/2025).

[58] Jacqueline Wong et al. “Supporting Self-Regulated Learning in Online Learning Environments and MOOCs: A Systematic Review”. In: *International Journal of Human–Computer Interaction* 35.4-5 (Mar. 2019). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10447318.2018.1543084, pp. 356–373. ISSN: 1044-7318. DOI: 10.1080/10447318.2018.1543084. URL: https://doi.org/10.1080/10447318.2018.1543084 (visited on 06/14/2021).

[59] Ruiwei Xiao et al. “A Preliminary Analysis of Students’ Help Requests with an LLM-powered Chatbot when Completing CS1 Assignments”. en_ca. In: (June 2024). Publisher: CSEDM’24: 8th Educational Data Mining in Computer Science Education (CSEDM) Workshop. URL: http://hdl.handle.net/1807/139381 (visited on 04/03/2025).

[60] Anjali Khurana, Hariharan Subramonyam, and Parmit K Chilana. “Why and When LLM-Based Assistants Can Go Wrong: Investigating the Effectiveness of Prompt-Based Interactions for Software Help-Seeking”. In: *Proceedings of the 29th International Conference on Intelligent User Interfaces*. IUI ’24. New York, NY, USA: Association for Computing Machinery, Apr. 2024, pp. 288–303. ISBN: 9798400705083. DOI: 10.1145/3640543.3645200. URL: https://doi.org/10.1145/3640543.3645200 (visited on 04/20/2025).

[61] Yizhou Fan et al. “Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance”. en. In: *British Journal of Educational Technology* 56.2 (2025). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.13544, pp. 489–530. ISSN: 1467-8535. DOI: 10.1111/bjet.13544. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/bjet.13544 (visited on 04/20/2025).

[62] Shamya Karumbaiah, Jaclyn Ocumpaugh, and Ryan S. Baker. “Context Matters: Differing Implications of Motivation and Help-Seeking in Educational Technology”. en. In: *International Journal of Artificial Intelligence in Education* 32.3 (Sept. 2022), pp. 685–724. ISSN: 1560-4306. DOI: 10.1007/s40593-021-00272-0. URL: https://doi.org/10.1007/s40593-021-00272-0 (visited on 01/31/2025).

A

Data Management Plan

Plan Overview

A Data Management Plan created using DMPonline

Title: Using LLMs for support in high school programming education

Creator: Thom van der Velden

Contributor: Marcus Specht, Manuel Valle Torre

Affiliation: Delft University of Technology

Template: TU Delft Data Management Plan template (2021)

ID: 157911

Start date: 26-08-2024

End date: 17-04-2025

Last modified: 10-09-2024

Using LLMs for support in high school programming education

0. Administrative questions

1. Name of data management support staff consulted during the preparation of this plan.

My faculty data steward, Santosh Ilamparuthi has reviewed this DMP on 09/03/2024.

2. Date of consultation with support staff.

2024-09-03

I. Data description and collection or re-use of existing data

3. Provide a general description of the type of data you will be working with, including any re-used data:

Type of data	File format(s)	How will data be collected (for re-used data: source and terms of use)?	Purpose of processing	Storage location	Who will have access to the data
Anonymised programming data	text files	notebook logger	to be able to see what students are doing while making the programming assignments	Faculty managed server / project drive	The project team
Anonymised LLM conversation with students	text files	LLM logger	to be able to see what students are asking to the chat while making the programming assignments	Faculty managed server / project drive	The project team
notebooks with programming exercises	Jupyter Notebook	made by myself	exercises that students are doing	Faculty managed server / project drive	The project team / students
Informed consent both from parents and students	paper	on paper signed by parents and students	to get consent of the students who are participating	Manuel	The project team
Survey data	Questions with likert scale	Qualtrics	to ask students their trust and expectations of the LLM throughout the experiment	Qualtrics	The project team

4. How much data storage will you require during the project lifetime?

- < 250 GB

II. Documentation and data quality

5. What documentation will accompany data?

- Methodology of data collection
- README file or other documentation explaining how data is organised

III. Storage and backup during research process

6. Where will the data (and code, if applicable) be stored and backed-up during the project lifetime?

- Project Storage at TU Delft

IV. Legal and ethical requirements, codes of conduct

7. Does your research involve human subjects or 3rd party datasets collected from human participants?

- Yes

It involves high school students who are doing a programming course

8A. Will you work with personal data? (information about an identified or identifiable natural person)

If you are not sure which option to select, first ask your Faculty Data Steward for advice. You can also check with the privacy website . If you would like to contact the privacy team: privacy-tud@tudelft.nl, please bring your DMP.

- Yes

Direct data collection is not personal data, but the final grades of the students will be used to evaluate their performance. These grades are personal, however only the teacher (me) will be able to link the grades to the data of the students.

8B. Will you work with any other types of confidential or classified data or code as listed below? (tick all that apply)

If you are not sure which option to select, ask your Faculty Data Steward for advice.

- Yes, I work with other types of confidential or classified data (or code) - please explain below

student grade data is used which is confidential because students would not want everyone to know their grades

9. How will ownership of the data and intellectual property rights to the data be managed?

For projects involving commercially-sensitive research or research involving third parties, seek advice of your Faculty Contract Manager when answering this question. If this is not the case, you can use the example below.

Data will be owned by TU Delft since it is funded by the university. Confidential part of the data will not be published (actual grades).

10. Which personal data will you process? Tick all that apply

- Data collected in Informed Consent form (names and email addresses)
- Signed consent forms
- Other types of personal data - please explain below

apart from consent we collect student grades and that is all

11. Please list the categories of data subjects

students in high school in a havo 4 Informatica course

12. Will you be sharing personal data with individuals/organisations outside of the EEA (European Economic Area)?

- No

15. What is the legal ground for personal data processing?

- Informed consent

16. Please describe the informed consent procedure you will follow:

Beste ouder / verzorger / leerling,

Ik ben leraar op het CLD van de 4 havo Informatica klas en ik ben op dit moment bezig met het laatste jaar van mijn master Informatica op de TU Delft. Voor deze master ben ik bezig met een scriptie over de invloed van AI chatbots op het leren programmeren van leerlingen. Hier is nog niet heel veel over bekend en het leek mij dus erg leuk om dat bij mijn eigen leerlingen te kunnen onderzoeken.

De leerlingen gaan over een paar weken beginnen met het leren programmeren en zij gaan dit doen in een omgeving waarbij een AI chatbot aanwezig is. Hier kunnen leerlingen vragen aan stellen als zij vastlopen en de chatbot zal hen dan proberen verder te helpen. Als dit niet lukt of als ze liever vragen aan mij stellen kan dat natuurlijk ook. Met mijn onderzoek wil ik erachter komen wat de invloed van deze chatbot is op het leren van de leerlingen. De omgeving waarin zij werken zal opslaan wat de leerlingen vragen aan de bot en zal tegelijkertijd ook opslaan wat de leerlingen typen tijdens het programmeren. Door achteraf naar deze data te kijken hoop ik verbanden te kunnen ontdekken tussen het uiteindelijke niveau van leerlingen en het gebruik van de chatbot, bijvoorbeeld: “Leerlingen die meer vragen stelden aan de chatbot haalden hogere cijfers op hun toetsen.”

Dit onderzoek heeft uiteraard geen invloed op de leerlingen. Zij zullen niks van dit onderzoek merken tijdens de lessen en hoe zij met de chatbot omgaan heeft geen invloed op hun cijfers tijdens de toetsen. Tijdens de toetsen zal de chatbot niet aanwezig zijn dus we zullen natuurlijk ook zonder chatbot oefenen. Na het onderzoek zal er geen enkele manier zijn waarop de leerling gekoppeld zou kunnen worden aan het onderzoek.

Zoals bij elke online activiteit is het risico van een data breuk aanwezig. Wij doen ons best om uw antwoorden vertrouwelijk te houden. We maken geen gebruik van derde partijen en we slaan alle data op onze TU Delft servers op. We minimaliseren de risico’s door de data anoniem te verzamelen en zo min mogelijk persoonlijke data op te slaan. Het enige wat we van de leerlingen zullen weten zijn hun username, waarmee ze een account in de online omgeving kunnen maken, en hun uiteindelijke resultaten. Deze resultaten zullen na het onderzoek niet meer te koppelen zijn aan een specifieke leerling.

Deelname aan dit onderzoek is natuurlijk volledig vrijwillig, als de leerling niet wilt meedoen aan dit onderzoek hoeft deze brief niet getekend te worden. Ook kan de leerling zich elk moment terugtrekken uit het onderzoek zonder reden op te geven als de leerling zich bedenkt. Als de leerling niet meedoet aan het onderzoek zal de leerling natuurlijk nog steeds met de les mee kunnen doen. De data van de betreffende leerling zal dan echter niet opgeslagen worden.

Als u meer informatie wilt over het onderzoek kunt u contact opnemen met mij.

Met vriendelijke groet,
Thom van der Velden

Zet alstublieft een kruisje in het passende vakje	Ja	Nee
1. Ik heb de informatie over het onderzoek gedateerd 07-10-2024 tot 20-02-2025 gelezen en begrepen, of deze is aan mij voorgelezen.	☐	☐
2. Ik doe vrijwillig mee aan dit onderzoek, en ik begrijp dat ik kan weigeren dat ik meedoe en dat ik me op elk moment kan terugtrekken uit de studie, zonder een reden op te hoeven geven.	☐	☐
3. Ik begrijp dat deelname aan het onderzoek betekent dat de omgeving waarin ik zal werken, zal opslaan wat ik vraag aan de chatbot en tegelijkertijd ook zal opslaan wat ik typ tijdens het programmeren. Deze data zal vervolgens gebruikt worden om de invloed van deze chatbot op de leerlingen te onderzoeken.	☐	☐
4. Ik begrijp dat de studie aan het einde van blok 3 (midden februari) eindigt.	☐	☐
5. Ik begrijp dat de persoonlijke informatie die over mij verzameld wordt en mij kan identificeren, zoals mijn naam, niet gedeeld worden buiten het studieteam.	☐	☐
6. Ik begrijp dat mijn examencijfers voor allebei mijn programmeer examens gebruikt zullen worden in de evaluatie van het onderzoek. Ik begrijp dat mijn interactie met de chatbot op geen enkele manier invloed zal hebben op mijn examencijfers. Ik begrijp dat de uiteindelijke gepubliceerde dataset geen individuele cijfers van leerlingen zal hebben.	☐	☐
7. Ik begrijp dat de persoonlijke data die over mij verzameld wordt, vernietigd wordt als het onderzoek klaar is, dat zal hoogstwaarschijnlijk rond het einde van dit schooljaar zijn.	☐	☐
8. Ik begrijp dat na het onderzoek de geanonimiseerde informatie gebruikt zal worden voor de onderzoeker (Thom) zijn thesis report. Ook is er de mogelijkheid dat dit onderzoek gepubliceerd zal worden.	☐	☐
9. Ik geef toestemming dat de geanonimiseerde data (vragen gesteld aan de chatbot en getypte code) die over mij verzameld worden, gearchiveerd worden in een 4TU repository opdat deze gebruikt kunnen worden voor toekomstig onderzoek en onderwijs.	☐	☐

Handtekeningen
Ik, de leerling, verklaar dat ik alle informatie in deze brief heb begrepen en hiermee akkoord ben. Ik verklaar dat ik mijn instemming voor dit onderzoek vrijwillig heb gegeven. _____ _____ _____ Naam deelnemer Handtekening Datum
Ik, de wettelijke vertegenwoordiger (ouder / verzorger), verklaar dat de informatie en het instemmingsformulier aan de potentiële deelnemer correct zijn voorgedragen, en dat hij/zij de kans heeft gekregen om vragen te stellen. Ik verklaar dat de potentiële deelnemer zijn/haar instemming vrijwillig heeft gegeven. _____ _____ _____ Naam wettelijke vertegenwoordiger Handtekening Datum
Ik, de onderzoeker, verklaar dat ik de informatie en het instemmingsformulier correct aan de potentiële deelnemer heb voorgedragen en, naar het beste van mijn vermogen, heb verzekerd dat de deelnemer begrijpt waar hij/zij vrijwillig mee instemt. _____ _____ _____ Naam onderzoeker Handtekening Datum

This is the form that will be signed by parents and students.

17. Where will you store the signed consent forms?

- Other - please explain below

We will store the consent forms on paper inside of a folder in a cabinet that is locked.

18. Does the processing of the personal data result in a high risk to the data subjects?

If the processing of the personal data results in a high risk to the data subjects, it is required to perform a Data Protection Impact Assessment (DPIA). In order to determine if there is a high risk for the data subjects, please check if any of the options below that are applicable to the processing of the personal data during your research (check all

that apply).
If two or more of the options listed below apply, you will have to complete the DPIA. Please get in touch with the privacy team: privacy-tud@tudelft.nl to receive support with DPIA.
If only one of the options listed below applies, your project might need a DPIA. Please get in touch with the privacy team: privacy-tud@tudelft.nl to get advice as to whether DPIA is necessary.
If you have any additional comments, please add them in the box below.

- Sensitive personal data
- Evaluation or scoring

19. Did the privacy team advise you to perform a DPIA?

- No

We asked and they agreed we do not need to perform a DPIA.

22. What will happen with personal research data after the end of the research project?

- Personal research data will be destroyed after the end of the research project

V. Data sharing and long-term preservation

27. Apart from personal data mentioned in question 22, will any other data be publicly shared?

- All other non-personal data (and code) underlying published articles / reports / theses

29. How will you share research data (and code), including the one mentioned in question 22?

- All anonymised or aggregated data, and/or all other non-personal data will be uploaded to 4TU.ResearchData with public access

30. How much of your data will be shared in a research data repository?

- < 100 GB

31. When will the data (or code) be shared?

- At the end of the research project

32. Under what licence will be the data/code released?

- CC BY

VI. Data management responsibilities and resources

33. Is TU Delft the lead institution for this project?

- Yes, the only institution involved

34. If you leave TU Delft (or are unavailable), who is going to be responsible for the data resulting from this project?

Marcus Specht (Professor)(m.m.specht@tudelft.nl) and
Manuel Valle Torre (PhD) (m.valletorre@tudelft.nl)

35. What resources (for example financial and time) will be dedicated to data management and ensuring that data will be FAIR (Findable, Accessible, Interoperable, Re-usable)?

no additional resources are required

B

Human Research Ethics Committee

Delft University of Technology
HUMAN RESEARCH ETHICS
CHECKLIST FOR HUMAN RESEARCH
(Version January 2022)

IMPORTANT NOTES ON PREPARING THIS CHECKLIST

1. An HREC application should be submitted for every research study that involves human participants (as Research Subjects) carried out by TU Delft researchers
2. Your HREC application should be submitted and approved **before** potential participants are approached to take part in your study
3. All submissions from Master's Students for their research thesis need approval from the relevant Responsible Researcher
4. The Responsible Researcher must indicate their approval of the completeness and quality of the submission by signing and dating this form OR by providing approval to the corresponding researcher via email (included as a PDF with the full HREC submission)
5. There are various aspects of human research compliance which fall outside of the remit of the HREC, but which must be in place to obtain HREC approval. These often require input from internal or external experts such as Faculty Data Stewards, Faculty HSE advisors, the TU Delft Privacy Team or external Medical research partners.
6. You can find detailed guidance on completing your HREC application here
7. Please note that incomplete submissions (whether in terms of documentation or the information provided therein) will be returned for completion **prior to any assessment**
8. If you have any feedback on any aspect of the HREC approval tools and/or process you can leave your comments here

I. Applicant Information

PROJECT TITLE:	**Using LLMs for support in high school programming education**
Research period: *Over what period of time will this specific part of the research take place*	**26-08-2024 until 17-04-2025**
Faculty:	**EEMCS**
Department:	**Web Information Systems**
Type of the research project: *(Bachelor's, Master's, DreamTeam, PhD, PostDoc, Senior Researcher, Organisational etc.)*	**Master's**
Funder of research: *(EU, NWO, TUD, other – in which case please elaborate)*	**TUD**
Name of Corresponding Researcher: *(If different from the Responsible Researcher)*	**Thom van der Velden**
E-mail Corresponding Researcher: *(If different from the Responsible Researcher)*	**thomvanderveld@tudelft.nl**
Position of Corresponding Researcher: *(Masters, DreamTeam, PhD, PostDoc, Assistant/ Associate/ Full Professor)*	**Masters**
Name of Responsible Researcher: ***Note:*** *all student work must have a named Responsible Researcher to approve, sign and submit this application*	**Manuel Valle Torre**
E-mail of Responsible Researcher: *Please ensure that an institutional email address (**no Gmail, Yahoo, etc.**) is used for all project documentation/ communications including Informed Consent materials*	**m.valletorre@tudelft.nl**
Position of Responsible Researcher : *(PhD, PostDoc, Associate/ Assistant/ Full Professor)*	**PhD**

II. Research Overview

NOTE: *You can find more guidance on completing this checklist here*

a) Please summarise your research very briefly (100-200 words)

What are you looking into, who is involved, how many participants there will be, how they will be recruited and what are they expected to do?

Add your text here – (please avoid jargon and abbrevations)
I will research the use of LLMs for high school students when they are learning to program. I have a class of 22 students who are going to learn programming in Python. This is part of their high school curriculum. They are learning from jupyter notebooks and they can ask questions to a LLM that can support them with their programming. These conversations will be logged so that we can analyze the interactions afterwards. The students are around 15/16 years old.

b) **If your application is an additional project** related to an existing approved HREC submission, please provide a brief explanation including the existing relevant HREC submission number/s.

Add your text here – (please avoid jargon and abbrevations)

c) **If your application is a simple extension of, or amendment to,** an existing approved HREC submission, you can simply submit an HREC Amendment Form as a submission through LabServant.

III. Risk Assessment and Mitigation Plan

NOTE: *You can find more guidance on completing this checklist here*

Please complete the following table in full for all points to which your answer is "yes". Bear in mind that the vast majority of projects involving human participants as Research Subjects also involve the collection of **Personally Identifiable Information (PII)** and/or **Personally Identifiable Research Data (PIRD)** which may pose potential risks to participants as detailed in Section G: Data Processing and Privacy below.

To ensure alighment between your risk assessment, data management and what you agree with your Research Subjects you can use the last two columns in the table below to refer to specific points in your Data Management Plan (DMP) and Informed Consent Form (ICF) – **but this is not compulsory**.

It's worth noting that **you're much more likely to need to resubmit your application if you neglect to identify potential risks**, than if you identify a potential risk and demonstrate how you will mitigate it. If necessary, the HREC will always work with you and colleagues in the Privacy Team and Data Management Services to see how, if at all possible, your research can be conducted.

			If YES please complete the Risk Assessment and Mitigation Plan columns below.		*Please provide the relevant reference #*	
ISSUE	**Yes**	**No**	**RISK ASSESSMENT – what risks could arise?** *Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!*	**MITIGATION PLAN – what mitigating steps will you take?** *Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.*	**DMP**	**ICF**
A: Partners and collaboration						
1. Will the research be carried out in collaboration with additional organisational partners such as: • One or more collaborating research and/or commercial organisations • Either a research, or a work experience internship provider[1] [1] *If yes, please include the graduation agreement in this application*		**No**				
2. Is this research dependent on a Data Transfer or Processing Agreement with a collaborating partner or third party supplier? *If yes please provide a copy of the signed DTA/DPA*		**No**				
3. Has this research been approved by another (external) research ethics committee (e.g.: HREC and/or MREC/METC)? *If yes, please provide a copy of the approval (if possible) and summarise any key points in your Risk Management section below*		**No**				
B: Location						

			If YES please complete the Risk Assessment and Mitigation Plan columns below.		*Please provide the relevant reference #*	
ISSUE	**Yes**	**No**	**RISK ASSESSMENT – what risks could arise?** *Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!*	**MITIGATION PLAN – what mitigating steps will you take?** *Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.*	**DMP**	**ICF**
4. Will the research take place in a country or countries, other than the Netherlands, within the EU?		**No**				
5. Will the research take place in a country or countries outside the EU?		**No**				
6. Will the research take place in a place/region or of higher risk – including known dangerous locations (in any country) or locations with non-democratic regimes?		**No**				
C: Participants						
7. Will the study involve participants who **may** be vulnerable and possibly (legally) unable to give informed consent? (e.g., children below the legal age for giving consent, people with learning difficulties, people living in care or nursing homes).	**Yes**		Given that the participants are minors, there is a risk that they might not fully understand the implications of the study or their rights within it. This could lead to misunderstandings about what participation involves or what they are agreeing to. Minors may feel pressured to agree to participate if they perceive it as a requirement or if they feel that refusal could result in negative consequences, such as disappointing their teacher or being singled out.	Parents will also need to sign an informed consent form so that we will be certain everyone knows what they are agreeing to. We will reinforce the voluntary nature of the study in all interactions with the students and their parents/guardians. Make it clear that participation is not tied to academic evaluation or other school-related outcomes. After we have started the course the students won't even notice they are participating in the research.		
8. Will the study involve participants who **may** be vulnerable under specific circumstances and in specific contexts, such as victims and witnesses of violence, including domestic violence; sex workers; members of minority groups, refugees, irregular migrants or dissidents?		**No**				
9. Are the participants, outside the context of the research, in a dependent or subordinate position to the investigator (such as own children, own students or employees of either TU Delft and/or a collaborating partner organisation)? *It is essential that you safeguard against possible adverse consequences of this situation (such as allowing a student's failure to participate to your satisfaction to affect your evaluation of their coursework).*	**Yes**		Students might feel obligated to participate in the study or to interact with the LLM in a certain way because I am their teacher and responsible for evaluating their coursework. They may fear negative consequences if they choose not to participate or if they don't perform to the expected standard in the research context. The students may tailor their responses or behavior to what they believe I want to see, rather than acting	Ensure that students and their parents or guardians understand that participation in the study is entirely voluntary and that their decision to participate or not will have no impact on their grades or standing in the class. Implement measures to ensure that all data collected is anonymized, meaning I will not be able to link		

			If YES please complete the Risk Assessment and Mitigation Plan columns below.		*Please provide the relevant reference #*	
ISSUE	**Yes**	**No**	**RISK ASSESSMENT – what risks could arise?** *Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!*	**MITIGATION PLAN – what mitigating steps will you take?** *Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.*	**DMP**	**ICF**
			naturally. This could compromise the validity of the data collected.	individual responses or behaviors to specific students. Inform students that their individual responses and interactions with the LLM will not be visible to me in a way that can identify them personally. We will only link the grades and the collected data for analysis after the grades are collected. The data collected will not affect their grades.		
10. Is there a high possibility of re-identification for your participants? (e.g., do they have a very specialist job of which there are only a small number in a given country, are they members of a small community, or employees from a partner company collaborating in the research? Or are they one of only a handful of (expert) participants in the study?		**No**				
D: Recruiting Participants						
11. Will your participants be recruited through your own, professional, channels such as conference attendance lists, or through specific network/s such as self-help groups		**No**				
12. Will the participants be recruited or accessed in the longer term by a (legal or customary) gatekeeper? (e.g., an adult professional working with children; a community leader or family member who has this customary role – within or outside the EU; the data producer of a long-term cohort study)		**No**				
13. Will you be recruiting your participants through a crowd-sourcing service and/or involve a third party data-gathering service, such as a survey platform?		**No**				
14. Will you be offering any financial, or other, remuneration to participants, and might this induce or bias participation?		**No**				
E: Subject Matter *Research related to medical questions/health may require special attention. See also the website of the CCMO before contacting the HREC.*						
15. Will your research involve any of the following: • Medical research and/or clinical trials • Invasive sampling and/or medical imaging • Medical and *In Vitro Diagnostic Medical* Devices Research		**No**				
16. Will drugs, placebos, or other substances (e.g., drinks, foods, food or drink constituents, dietary supplements) be administered to the study participants? *If yes see here to determine whether medical ethical approval is required*		**No**				

			If YES please complete the Risk Assessment and Mitigation Plan columns below.		*Please provide the relevant reference #*	
ISSUE	**Yes**	**No**	**RISK ASSESSMENT – what risks could arise?** *Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!*	**MITIGATION PLAN – what mitigating steps will you take?** *Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.*	**DMP**	**ICF**
17. Will blood or tissue samples be obtained from participants? *If yes see here to determine whether medical ethical approval is required*		**No**				
18. Does the study risk causing psychological stress or anxiety beyond that normally encountered by the participants in their life outside research?		**No**				
19. Will the study involve discussion of personal sensitive data which could put participants at increased legal, financial, reputational, security or other risk? (e.g., financial data, location data, data relating to children or other vulnerable groups) *Definitions of sensitive personal data, and special cases are provided on the TUD Privacy Team website.*		**No**				
20. Will the study involve disclosing commercially or professionally sensitive, or confidential information? (e.g., relating to decision-making processes or business strategies which might, for example, be of interest to competitors)		**No**				
21. Has your study been identified by the TU Delft Privacy Team as requiring a Data Processing Impact Assessment (DPIA)? *If yes please attach the advice/ approval from the Privacy Team to this application*		**No**				
22. Does your research investigate causes or areas of conflict? *If yes please confirm that your fieldwork has been discussed with the appropriate safety/security advisors and approved by your Department/Faculty.*		**No**				
23. Does your research involve observing illegal activities or data processed or provided by authorities responsible for preventing, investigating, detecting or prosecuting criminal offences *If so please confirm that your work has been discussed with the appropriate legal advisors and approved by your Department/Faculty.*		**No**				
F: Research Methods						
24. Will it be necessary for participants to take part in the study without their knowledge and consent at the time? (e.g., covert observation of people in non-public places).		**No**				
25. Will the study involve actively deceiving the participants? (For example, will participants be deliberately falsely informed, will information be withheld from them or will they be misled in such a way that they are likely to object or show unease when debriefed about the study).		**No**				

			If YES please complete the Risk Assessment and Mitigation Plan columns below.		*Please provide the relevant reference #*	
ISSUE	**Yes**	**No**	**RISK ASSESSMENT – what risks could arise?** *Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!*	**MITIGATION PLAN – what mitigating steps will you take?** *Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.*	**DMP**	**ICF**
26. Is pain or more than mild discomfort likely to result from the study? And/or could your research activity cause an accident involving (non-) participants?		**No**				
27. Will the experiment involve the use of devices that are not 'CE' certified? *Only, if 'yes': continue with the following questions:*		**No**				
• Was the device built in-house?						
• Was it inspected by a safety expert at TU Delft? *If yes, please provide a signed **device report***						
• If it was not built in-house and not CE-certified, was it inspected by some other, qualified authority in safety and approved? *If yes, please provide records of the inspection*						
28. Will your research involve face-to-face encounters with your participants and if so how will you assess and address Covid considerations?	**Yes**		It could be possible that Covid would be spread	They are high school students, the school has rules for Covid that we will follow throughout the whole experiment		
29. Will your research involve **either:** a) "big data", combined datasets, new data-gathering or new data-merging techniques which might lead to re-identification of your participants **and/or** b) artificial intelligence or algorithm training where, for example biased datasets could lead to biased outcomes?	**Yes**		The study involves logging all student interactions with the LLM and tracking their problem-solving processes. While the data collected is educational and not overtly sensitive, the combination of detailed interaction logs, even when anonymized, might present a risk of re-identification if not handled properly. If specific patterns of behavior or unique identifiers were improperly managed, there could be a potential risk that a student's identity could be inferred from the data. It is possible that the LLM introduces biases.	Make sure the data is strictly anonymized and instruct students to not put personal information in the code cells and the chatbot. For example: make sure they do not use their last name as the name of a variable. If we would spot something like this we would remove it from the data to make sure it is still properly anonymized. Since all the interactions with the LLM will be about introductory programming we do not expect that inherent biases of the LLM will affect our outcomes.		
G: Data Processing and Privacy						
30. Will the research involve collecting, processing and/or storing any directly identifiable PII (Personally Identifiable Information) including name or email address that will be used for administrative purposes only? (eg: obtaining Informed Consent or disbursing remuneration)	**Yes**		There could be a possible data breach if data storage is not properly secured PII could be accidentally shared with unauthorized parties through for example email misdirection	We collect the consent forms on paper. These papers will be put in a folder in a cabinet with a secure lock so that unauthorized people won't be able to access the consent forms. Since we ask the student and parents to sign on paper this risk is mitigated.		

			If YES please complete the Risk Assessment and Mitigation Plan columns below.		*Please provide the relevant reference #*	
ISSUE	**Yes**	**No**	**RISK ASSESSMENT – what risks could arise?** *Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!*	**MITIGATION PLAN – what mitigating steps will you take?** *Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.*	**DMP**	**ICF**
31. Will the research involve collecting, processing and/or storing any directly or indirectly identifiable PIRD (Personally Identifiable Research Data) including videos, pictures, IP address, gender, age etc and **what other Personal Research Data** (including personal or professional views) will you be collecting?	**Yes**		We will be collecting students their exam grades for this programming course. Risks: Unauthorized access	I will be the only person that can link the grades to the students data, after that it is completely anonymous. The data collected is just programming logs and questions about programming so there is no more personal information than the one for administration.		
32. Will this research involve collecting data from the internet, social media and/or publicly available datasets which have been originally contributed by human participants		**No**				
33. Will your research findings be published in one or more forms in the public domain, as e.g., Masters thesis, journal publication, conference presentation or wider public dissemination?	**Yes**		No risks, data will be published without identifiable information			
34. Will your research data be archived for re-use and/or teaching in an open, private or semi-open archive?	**Yes**		If necessary can be obfuscated			

H: More on Informed Consent and Data Management

NOTE: *You can find guidance and templates for preparing your Informed Consent materials) here*

Your research involves human participants as Research Subjects if you are recruiting them or actively involving or influencing, manipulating or directing them in any way in your research activities. This means you must seek informed consent and agree/ implement appropriate safeguards regardless of whether you are collecting any PIRD.

Where you are also collecting PIRD, and using Informed Consent as the legal basis for your research, you need to also make sure that your IC materials are clear on any related risks and the mitigating measures you will take – including through responsible data management.

Got a comment on this checklist or the HREC process? You can leave your comments here

IV. Signature/s

Please note that by signing this checklist list as the sole, or Responsible, researcher you are providing approval of the completeness and quality of the submission, as well as confirming alignment between GDPR, Data Management and Informed Consent requirements.

Name of Corresponding Researcher (if different from the Responsible Researcher) (print)
Thom van der Velden

Signature of Corresponding Researcher:

Date: 11/sep/2024

Name of Responsible Researcher (print)
Manuel Valle Torre

Signature (or upload consent by mail) Responsible Researcher:

Date: 11/Sep/2024

V. Completing your HREC application

Please use the following list to check that you have provided all relevant documentation

Required:

- **Always:** This completed HREC checklist
- **Always:** A data management plan (reviewed, where necessary, by a data-steward)
- **Usually:** A complete Informed Consent form (including Participant Information) and/or Opening Statement (for online consent)

Please also attach any of the following, if relevant to your research:

Document or approval	Contact/s
Full Research Ethics Application	After the assessment of your initial application **HREC will let you know if and when you need to submit additional information**
Signed, valid Device Report	Your Faculty HSE advisor
Ethics approval from an external Medical Committee	TU Delft Policy Advisor, Medical (Devices) Research
Ethics approval from an external Research Ethics Committee	Please append, if possible, with your submission
Approved Data Transfer or Data Processing Agreement	Your Faculty Data Steward and/or TU Delft Privacy Team
Approved Graduation Agreement	Your Master's thesis supervisor
Data Processing Impact Assessment (DPIA)	TU Delft Privacy Team
Other specific requirement	Please reference/explain in your checklist and append with your submission