DELFT UNIVERSITY OF TECHNOLOGY

MASTERS THESIS

Enhancing Programming Education Through LLM-Based Learning Analytics and Interventions

Author:
Colin Busropan
Student number:
4658272

Thesis Supervisor:
Prof. Dr. M.M. SPECHT
Daily Supervisor:
MSc. Manuel Valle Torre

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Web Information Systems Group
Software Technology

July 10, 2024

TU Delft

Declaration of Authorship

I, Colin Busropan, declare that this thesis titled, "Enhancing Programming Education Through LLM-Based Learning Analytics and Interventions" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: 2024-07-10

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

Electrical Engineering, Mathematics and Computer Science
Software Technology

Master of Science

Enhancing Programming Education Through LLM-Based Learning Analytics and Interventions

by Colin Busropan

The rising number of students in computer science presents a challenge for educators to analyse student work and identify problem areas at scale. Learning analytics systems often fall short in offering detailed, actionable insights that educators can use to enhance their teaching. To address this, we propose a system leveraging Large Language Models to analyse programming submissions and generate actionable analytics. The study focused on two primary research questions regarding the accuracy of LLMs in classifying programming submissions and identifying common issues, and educators' perceptions of the usefulness of the system. Through a system evaluation and focus group, we found that LLMs can analyse SQL assignment submissions with reasonable accuracy and can identify common issues. Educators found the insights potentially useful but noted areas for improvement such as the need for concrete statistics, accurate lists of submissions with specific issues, and more readable reports. Future research should address these implications for the system, evaluate the system in other courses with different programming languages, and involve stakeholders directly related to the course.

Acknowledgements

I would like to express my deepest gratitude to everyone who has supported me throughout the journey of completing this thesis.

First and foremost, I would like to thank Professor Marcus Specht and MSc. Manuel Valle Torre for their supervision and guidance throughout my research. Your expertise and support have been immensely valuable for me. I am also grateful to Professor Sicco Verwer for being part of my thesis committee. Special thanks also go to the participants of the pilot study and focus group for their time and the knowledge they provided to the project.

I am forever grateful to my parents for always supporting me and having faith in me. You are my main source of motivation and inspiration. I also want to express my appreciation to my peers, Safouane, Karthik, and Rebecca, in the research group for exchanging ideas throughout the thesis. Throughout the Master, I also had the pleasure of working with Otte and Suhaib on various projects, two excellent teammates with whom I shared a lot of fun outside the study as well. Finally, I would like to thank Dibyendu, Henry, Ian, Matteo, and Yash, who have been there since the start of my computer science journey. Your friendship and support have meant a lot to me.

Contents

List of Figures

List of Tables

List of Abbreviations

LA	**L**earning **A**nalytics
LLM	**L**arge **L**anguage **M**odel
AI	**A**rtificial **I**ntelligence
AIED	**A**rtificial **I**ntelligence in **Ed**ucation
GenAI	**Gen**erative **A**rtificial **I**ntelligence
NLP	**N**atural **L**anguage **P**rocessing
SQL	**S**tructured **Q**uery **L**anguage
GPT	**G**enerative **P**re-trained **T**ransformer
CNN	**C**onvolutional **N**eural **N**etwork
UI	**U**ser **I**nterface
TA	**T**eaching **A**ssistant

Chapter 1

Introduction

In computer science education, high student enrolment coupled with limited faculty resources has created significant challenges for educational institutions. The demand for skilled programmers and the growing interest in computer science has led to a boom in the number of students pursuing degrees in computer science and related fields [1]. This growth has outpaced the availability of faculty and teaching resources [2], necessitating a need for solutions and systems that can support both students and educators effectively.

Evaluating student performance and providing timely, personalised feedback are critical aspects of education. In traditional classroom settings, educators can interact directly with their students, providing personal guidance and support. However, this approach becomes infeasible with growing class sizes. To address this issue, many automated systems have been developed that support educators with the large number of students.

Learning Analytics (LA) has emerged as a field of study to address these problems. Learning analytics is "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs" [3]. The goal of LA is to understand and optimise learning processes and environments. By leveraging data from sources such as Learning Management Systems (LMS), learning analytics can provide insights into student performance, learning behaviours, and more. These insights can guide and inform teachers in improving their course or help them identify at-risk students.

However, learning analytics faces several challenges in programming education. Assessing a program is considerably more complex than verifying the functional correctness and as a result many different tools exist that go beyond this measure [4]. Program efficiency, behaviour, and readability, among many other features, are relevant for evaluation. However, these tools often require significant setup for a course or are not available for wider use, making it difficult to integrate them into different educational environments [5] [6]. Without assessing and getting feedback on these relevant features, students do not get a complete assessment of their work. Additionally, while learning analytics can provide valuable insights, they are often not actionable. Educators may receive indications of students at-risk by a learning analytics system, but they lack concrete suggestions for interventions to help these students or address their issues [7][8].

The recent advancements in artificial intelligence, particularly the rise of Large Language Models (LLMs) like ChatGPT [9], present a promising opportunity to enhance learning analytics in programming education and address these challenges. Trained on large amounts of data, these models can process and generate human-like text, while also handling unstructured data such as programming submissions [10]. Research has shown that LLMs can solve a wide range of programming problems

[11], as well as analysing programs and providing feedback on it [12]. By integrating LLMs into learning management systems, educators could gain better and more actionable insights into their students performance, helping them in supporting their students.

1.1 Objective

This thesis aims to address the challenge of effectively analysing student programming submissions to provide actionable learning analytics in the context of programming education. Specifically, it focuses on the design and evaluation of a system that uses LLMs to analyse programming submissions, aggregate these analyses, and generate actionable insights for educators that can support both individual student learning and overall course improvement.

1.2 Research Questions

The thesis focuses on two key research questions:

- **RQ1: How can LLMs be utilised to analyse programming submissions and provide detailed learning analytics in programming education?**

  To address this question, a system was designed and evaluated through two sub-questions:

  - **Sub-question 1 (RQ1a): How accurately can LLMs classify student programming submissions into predefined categories?**
    This research question seeks to evaluate the performance of LLMs in categorising student programming submissions into specific categories. By measuring the accuracy of these classifications, we can determine the capability of LLMs in providing foundational data that will be subsequently used to generate learning analytics.
  - **Sub-question 2 (RQ1b): Can LLMs identify common issues and patterns in student programming submissions, and how reliable are these identifications?**
    This question aims to explore the common issues LLMs can find in student programming assignments and assess the reliability of these identifications.

- **RQ2: How do educators perceive the usefulness of LLM-generated learning analytics?**

  This research question explores educators' perspectives on the usefulness of LLM-generated learning analytics. By gathering feedback from educators, we aim to identify the strengths and potential areas for improvement.

1.3 Thesis Outline

The thesis is structured as follows. Chapter 2 provides an overview of the related work in learning analytics, LLMs, and their applications in education. Chapter 3 details the design and development of our proposed system that generates learning analytics. In Chapter 4, the system is evaluated based on various performance

metrics. Chapter 5 presents the findings from a focus group study with educators. Chapter 6 consolidates the three previous chapters, presents their implications, discusses the limitations, and suggests directions for future work. Finally, Chapter 7 presents a conclusion to the thesis.

Chapter 2

Related Work

This chapter reviews relevant literature and research on Learning Analytics and Large Language Models.

Section 2.1 introduces Learning Analytics, its objectives, and key research themes. Section 2.2 discusses LLMs, including their capabilities and limitations. Section 2.3 reviews the use of LLMs in education. Finally, Section 2.4 outlines prompt engineering strategies for effectively using LLMs.

2.1 Learning Analytics

Learning Analytics (LA) is "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimising learning and the environments in which it occurs" [3]. The goal of LA is to understand and optimise learning processes and environments.

Clow [13] describes a generic LA cycle with four key steps: identifying learners, collecting data, generating analytics, and implementing interventions.

The first step in the cycle is identifying learners, which may be students enrolled in the course in university, learners in a Massive Open Online Course, or others. The next step is generating and capturing data about or generated by the learners. This involves gathering data points related to learners and their interactions, which can include demographic information, online activity, and assessment data. The third step is processing the collected data into meaningful metrics or analytics that provide insights into the learning process. This can be done with dashboards, visualisations, comparisons with previous years, and many more ways. The final step is closing the loop by using these analytics to introduce interventions that improve the learning experience. This can be done by directly assisting learners, but also indirectly by for example informing the teacher what to address in future cohorts of a course. This last step is also said to be the greatest challenge and aspect that should be addressed more in LA research [7][8].

A systematic meta-review [14] of LA research was conducted in 2019, analysing 901 articles to identify key trends, research gaps, and future directions in the field. This review categorised LA in four major research topics: prediction of performance, decision support for teachers and learners, detection of behavioural patterns & learner modelling, and dropout prediction. Below is a short description of each topic:

- The prediction of performance focuses on developing models that can predict student performance to help identify students at risk, enabling timely intervention [15]. For example, this was done in a study with data from the Open University UK, in which AI was used to profile low-engagement students based on data from a virtual learning environment [16].

- The goal of decision support for teachers and learners research is to enable both teachers and students to make data-driven decisions that enhance the learning experience and outcomes. Most studies in the review present dashboards as the form of decision support, making use of input variables such as learning time, types of learning behaviours, and types of course material accessed.
- To detect behavioural patterns & model learners, clustering is the most common technique and researchers mostly look at how actively students are participating in online learning activities.
- The last major theme is prediction of retention and dropout of students. Most studies look at the final grade of students as the target variable for these predictions.

The authors of the review of LA found that the majority of LA publications were focused on concepts or frameworks and conducting proof-of-concept analyses, rather than performing actual data analysis. For future research in this area, they recommend that researchers should go beyond merely identifying patterns or reporting analytic results. Researchers should suggest follow-up actions or interventions that can improve teaching and learning processes.

A review [17] of 24 case studies on learning analytics interventions in higher education revealed several common themes and challenges. They showed that interventions most frequently focus on increasing students' study performance, offering personalised feedback, and improving student retention. The data used in these interventions often include students' online learning behaviours, study performance, demographics, and course selection information. The challenges identified in the studies cover a wide range of aspects. Evaluating or generalising the intervention is a challenge, and the issue of scalability is apparent for example when there are too many requests for help from students.

Research on learning dashboards, a 'single display that aggregates different indicators about learners, learning processes, and/or learning contexts into one or multiple visualisations' [18], is still relatively recent and exploratory. Existing studies aim to identify what data is meaningful to different stakeholders and how to present this data to support their understanding and decision-making processes. In a recent review on learning dashboards [18], authors found that most studies with an evaluation component intend to gather feedback to improve the dashboards, focusing on constructs such as usability, usefulness, and user satisfaction. They found that the data sources for most dashboards are interaction logs, such as clicks and interactions within an LMS. The second most common data source are learning artefacts used or produced by learners, such as the content of student work. Notably, only four of the reviewed works use purely text to present data in teacher dashboards, with most dashboards utilising bar charts and line graphs. A challenge they identify is determining the appropriate detail level of information displayed on the dashboard, to prevent users from becoming overwhelmed or confused by the large volume of information.

2.1.1 Process Analysis

Learning is a continuously changing process, and aggregating or counting the learner's actions is limited in its narrative power [19]. Using analytics of the learning process to enable timely assessment and support is an ongoing development [19]. Direct teacher observation into the process of making an assignment is often impractical

due to the scale of students and data points during the process. To address this, research in this area often utilises process mining and sequence mining to understand how students approach tasks, making it possible to identify patterns that could be used for personalised support.

Still, the lack of interventions is a problem as mentioned in one literature review [19]. They reviewed literature on the use of sequence analysis in learning analytics, identifying trends, techniques, and applications in educational settings. They observed that only a few reviewed articles introduced interventions based on patterns observed.

A study by Blikstein [20] demonstrates the feasibility and usefulness of using learning analytics to identify patterns and trajectories in students' programming practices. They use code snapshots to predict academic performance and identify learning hurdles, but they acknowledge that the required human labelling for a subset of the data to determine similarity between snapshots was laborious. Also, while they introduce some directions for what can be done based on the findings, they do not specify detailed interventions.

To track snapshots and collect data, researchers developed an IDE plugin [21]. They presented a toolkit and used it to create a dataset with snapshots of solutions for assignment tasks. Although the study does not explore use cases more in-depth, one of the proposed use cases is identifying common errors and providing personalised hints.

Analysing snapshots of the code was done in another study [22] aimed to assist students who make programming errors. While the approach is innovative, it requires constructing a list of typical errors and accompanying Abstract Syntax Tree. This requirement limits its ease of use and applicability across different tasks and programming languages.

In a different study [23], snapshot metadata, such as number of attempts, is used to predict student performance and identifying those in need of support. However, their system does not provide interventions on what the students should improve, only indicating that they need more attention.

The previously described works demonstrate innovative approaches to utilising learner data in programming education to support students, but their methods fail to provide detailed, actionable interventions or are limited in the generalisability across programming languages to allow for more widespread use.

2.1.2 Artificial Intelligence in Education

The field of AI in education (AIED) is closely related to LA and has similar goals and methodologies for enhancing educational outcomes. Assessment and Evaluation was found to be the most common use of AIED in higher education in a literature review in 2023 [24]. Its main benefit is reducing the time and effort spent by teachers on grading students. One example is a system using Dynamic Bayesian Networks that adapts to learners in formative assessments [25], which can give both the student and teachers a better estimate of the student's progress. AI has also enabled teachers to provide feedback to students using NLP techniques [26][27].

Intelligent Tutoring Systems (ITS) are also one of the most common applications of AIED. These systems use data to adapt to each student and provide them with personalised learning pathways [28].

Profiling and prediction of students' performance is also a key application of AIED, as well as applications of AI for the managing of student learning, which

focuses on supporting educators by providing insights, organisation, and data analysis to optimise and manage student learning experiences [24].

A recent study [29] uses BERT [30] for grading and classifying student submissions of SQL assignments. The authors fine-tuned BERT on a dataset consisting of 12,899 submissions [31] combined with internal data, splitting it into 70% for training and 30% for testing. They achieved a high balanced accuracy of 93% in the multiclass classification task, significantly outperforming the baseline of 37% that uses a convolutional neural network (CNN) [32].

The authors in the study with BERT use Captum [33] to visualise the importance of specific elements in the SQL statements that influence the model's decisions. For illustration, Figure 2.1 shows the explainability obtained from the BERT model using the Captum tool in the multiclass classification task, with two examples per class.

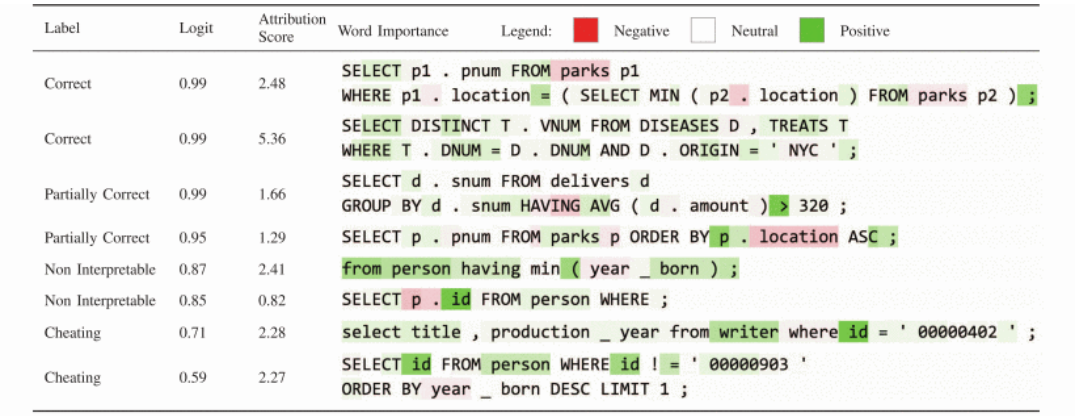

FIGURE 2.1: Explainability in the BERT study [29] using the Captum tool in the multiclass-classification task. The colour-coded parts represent the positive, neutral, or negative contribution to the classification decision, with darker colours showing stronger contributions. Two examples per class are shown.

The colour-coded parts represent the positive, neutral, or negative contribution to the classification decision, with darker colours showing stronger contributions. For example, 'id' is has a large contribution when classifying a statement as 'cheating'. While this tool provides insights for explainability, the visualisations are technical and may not be easily interpreted by all educators which could limit its accessibility and use in educational settings.

BERT was one of the most widely used LLMs before ChatGPT was introduced. In the next section we take a closer look into LLMs, specifically the GPT versions.

2.2 GPT-4 and Large Language Models

LLMs such as those in the Generative Pre-trained Transformer (GPT) series developed by OpenAI [34], are known for their ability to process and generate human-like text, which is based on extensive training on large sets of internet-sourced data. Introduced in 2023, GPT-4 [35] with its approximately 1 trillion parameters, has demonstrated an impressive proficiency in language understanding and generation. Furthermore, GPT-4 was shown to perform well on academic exams and showed state-of-the-art performance on various academic benchmarks [35].

A key advancement in the utilisation of LLMs has been their integration into ChatGPT. Launched in 2022, it allowed the public to experience this technology and showcase its ability to be utilised in a variety of tasks, for example in education for explaining concepts, and in creative industries for content creation. Moreover, the models performed well in code generation and programming tasks, enhancing the way people develop programs [11].

Despite these advancements, LLMs show inherent limitations that are important to acknowledge. Like any machine learning model, the data they are trained on can lead to the replication of unwanted biases present in the training data. Studies have shown that LLMs present stereotypes and biases, highlighting the need for careful use in applications where this could pose a risk [36].

Another limitation is the challenge of interpretability and transparency in LLMs. These models are often described as "black boxes" due to the difficulty in understanding how they arrive at certain outputs. This lack of transparency can be problematic in scenarios where understanding the decision-making process is crucial, such as in legal, medical applications, and in education. The complexity of these models also makes it hard to identify the source of errors or biases, making it hard to improve or correct them [36].

Additionally, LLMs still struggle with factual accuracy and "hallucinations", where the model confidently presents plausible but incorrect or nonsensical information. This issue is particularly concerning in applications that require high levels of accuracy, such as academic research or learning systems, and raises the need for human oversight and fact-checking when using LLM outputs in critical applications [36].

2.3 LLMs in Education

The use of LLMs in educational settings has been a topic of interest, as evidenced by various recent studies discussed in this section.

2.3.1 Grading and Assessment

Researchers have explored the potential of using LLMs for grading and assessment in educational contexts. Pinto et al. [37] focused on using ChatGPT for grading open-ended questions in technical training. The findings revealed a high level of agreement between the evaluations made by ChatGPT and subject matter experts. The potential of augmenting this process with expert-annotated grading rubrics was also highlighted, though the study noted its limited scope in question types.

Research detailed in [38] delved into the feasibility of LLM-assisted grading of handwritten physics solutions, integrating MathPix and GPT-4. This study, using a synthetic dataset generated by various GPT models, showed high agreement between LLM-assigned grades and human-assigned grades. However, the authors emphasised that while LLMs are promising for formative feedback, this approach is not yet reliable for high-stakes summative assessments, suggesting its best used as a supportive tool for human graders.

A further study explored ChatGPT's capabilities (both GPT-3.5 and GPT-4) in automatically grading a large set of student essays [39]. Here, GPT-4 demonstrated high accuracy, outperforming previous models, as evidenced by a lower Mean Squared Error when comparing it to human graders. The study suggested a need for a broader analysis assessing the effectiveness in other contexts.

Duolingo conducted research into the application of GPT-4 for the automated evaluation of discourse coherence in written language [40]. This study showed that GPT-4 could effectively rate discourse coherence, surpassing traditional NLP metrics. It also tested different prompt orders but found no significant difference between the GPT-4 configurations.

Another paper [41] described the development of an LLM-based software tool that automates the grading of text-based answers in educational settings. The key conclusion is the development and validation of the tool that automates and customises grading of text-based answers. The results indicated high agreement between the manual and LLM-based grading. The authors also suggest the possibility of enhanced accuracy by incorporating course materials in the context of the prompts.

2.3.2 Feedback Generation

A study [42] investigated the utility of GPT-4 in generating feedback directly for students in programming assignments, focusing on its accuracy in error detection and its capability to suggest correct functional code. The findings demonstrated a correctness classification of 84%, compared to 73% with GPT-3.5 on the same dataset [12]. Additionally, the completeness and accuracy of the feedback from GPT-4 was reported at 52%, significantly higher than the 31% achieved by GPT-3.5. The authors concluded that while directly presenting GPT-4 generated feedback to students is not recommended, the model could be used to assist teaching assistants in their roles.

In another publication [43], researchers compared ChatGPT's performance in providing student feedback with that of human instructors. ChatGPT could generate detailed feedback that coherently summarised student performance, though it was less reliable than instructor evaluations. They suggest improving performance by using few-shot prompting, which is a technique in which the LLM is presented with a few examples to learn from [44].

In related research [45], researchers introduced a new technique for generating hints for programming assignments, with a tutor model using GPT-4 and a student model using GPT-3.5. They use external tools to execute programs and extract useful symbolic information and then supply this information to GPT-4 for enhanced feedback generation, as well as chain-of-thought prompting [46] to enhance the responses from the LLM. The study's limitation, as noted by the authors, is the absence of real student participants in the experiment, suggesting this as a direction for future research.

A study introduced PyFiXV [47], which uses Codex [48], a general-purpose programming model, for generating feedback on Python syntax errors. Combining code correction and explanatory feedback with a validation mechanism showed promise in real-world datasets. The authors also acknowledged the potential for even better results with newer models like GPT4, and proposed conducting real-world studies in classrooms.

Another study [49] evaluated the use of GPT-3.5 for automated feedback generation in C# programming assignments for Polish computer science students. The results were positive, with significant improvements in student performance and positive ratings for the GPT-generated hints. However, concerns were raised about potential over-reliance on AI feedback, especially for more complex tasks.

2.3.3 Learning Analytics with LLMs

A recent paper [50] discusses integrating Generative Artificial Intelligence (GenAI), which includes LLMs, into learning analytics. They examine GenAI's role in Clow's [13] generic LA cycle with four key steps: identifying learners, collecting data, generating analytics, and implementing interventions.

1. **Identifying Learners**: As students increasingly use GenAI tools like ChatGPT, the concept of a "learner" is evolving, requiring a new understanding that focuses on human-AI collaboration.

2. **Collecting Data**: GenAI demonstrates potential for capturing unstructured learner data, as well as generating synthetic data. However, a challenge lies in identifying if students maintain the autonomy and control over their own learning [51] as GenAI can conduct various tasks for the learner.

3. **Generating Analytics**: The authors discuss GenAI's potential to enhance LA through the analysis of unstructured data, development of explanatory and interactive analytics, and the capability to generate multimodal content.

4. **Implementing Interventions**: The authors identify three key aspects GenAI can help in this underdeveloped step in LA: personalisation, adaptive support, and accessibility.

The paper concludes with future research suggestions focusing on hybrid human-AI learning and discusses ethical considerations for using GenAI in education, such as data privacy and biases.

In another study [52], the use of LLMs in the context of LA systems was examined. They investigated the potential of LLMs in generating customised learning tasks and assessments for a more personalised learning journey. Furthermore, the study underscores the importance of maintaining educator involvement in the automated feedback loop, ensuring a balance between AI-generated insights and human judgement. Additionally, the paper looks at how LLMs can be utilised across different phases of learning, from selecting appropriate materials to providing tailored feedback based on student outputs.

In conclusion, the studies in this section indicate that in a short time much research has been done towards integrating LLMs in educational settings, indicating that many researchers see potential in LLMs enhancing education. We briefly mentioned prompting, and in the following section, prompt engineering techniques will be highlighted.

2.4 Prompt Engineering

Prompt engineering is a crucial aspect to maximise the effectiveness of LLMs. The quality and relevance of the model's outputs can significantly be enhanced by carefully designing and structuring the input prompts. OpenAI [53] provides a detailed guide on constructing effective prompts to obtain desired outputs. This subsection will explore the core principles and strategies of prompt engineering, highlighting the guidelines from OpenAI and other sources [54][55].

Adopting a Persona

One suggested technique in prompt engineering is to instruct the model to adopt a specific persona. This can help the LLM select what should be included in the output and what relevant details to focus on based on the expertise that persona would have [56]. For example, one can instruct the model to "Act as a university professor" to elicit responses similar to what they would give.

Using Delimiters

To clearly distinguish between different elements of the prompt, delimiters such as quotation marks, XML tags, or other separators are recommended.

Summarising Long Documents

For long documents, breaking the document down into smaller sections and summarising each one sequentially can enhance comprehension and output quality. This also ensure all content fits in the context window of the LLM. LLMs have a context window, which is a limit to the size of the input and output of an LLM call. This limit differs per model. For instance, GPT-3.5 has a context window of 16k tokens, meaning it can handle up to 16,000 tokens in a single prompt interaction. In contrast, GPT-4 and GPT-4o support a much larger context window of 128k tokens. Tokens are essentially pieces of words that the model processes. In English, a token can be as short as a single character or as long as an entire word [57].

Few-Shot Prompting

Another valuable strategy is to provide examples of what the output should look like. This is referred to as few-shot prompting. Research [58] has shown improved performance across NLP tasks using this strategy, reducing the need for extensive fine-tuning or large task-specific datasets. However, this technique can be unstable as the choice of prompt format, training examples, and order of training examples can have a significant effect on the accuracy [59].

Chain-of-Thought Prompting

Chain-of-thought prompting is another strategy widely used and mention in literature. This technique involves breaking down complex tasks into intermediate steps, similar to showing your work in a math problem before arriving to your final answer. By guiding the model through a sequence of steps or thoughts, it can provide more accurate answers [46][60]. For example, instead of asking the model directly for the answer to a complex problem, you can prompt it to outline the steps it would take to arrive at the final response.

Additionally, OpenAI's guide [53] suggests using an inner monologue technique to hide the reasoning process. The idea is to put parts of the output that are meant to be hidden in a structured format that makes it easy to parse and subsequently hide from the user. For example, in a tutoring application it is essential to not give away the answer when trying to provide hints to a student. Using this technique can prevent this while also ensuring the chain-of-thought technique is used to improve reasoning.

Chapter 3

System Design

This chapter details the design of the system developed in this thesis. Section 3.1 outlines the system's objective. Section 3.2 explains the requirements analysis, detailing key considerations. Section 3.3 provides a comprehensive overview of the system design, Section 3.4 discusses the data flow, and Section 3.5 shows the prompts used to instruct the LLM. Section 3.6 describes the user interface design, and lastly, Section 3.7 presents a discussion of the system design.

3.1 Objective of the System

The objective of the system introduced in this thesis is to leverage LLMs to provide educators detailed learning analytics about the submissions of programming assignments that students make throughout the course. The goal is not to replace educators, but rather to assist them in their decision-making by analysing large quantities of data that would otherwise require significant time and effort to analyse in detail.

Specifically, the system aims to:

- Generate reports on common issues or patterns in the student submissions, across the entire course.
- Generate reports for each assignment, highlighting common issues or patterns observed. This can inform educators on how to improve the assignment or on issues that should be addressed in follow-up lessons.
- Generate reports for each student, highlighting common issues or patterns observed. This can inform educators on how to support individual students effectively.
- Analyse the process of a student making an assignment. By examining the sequence of submissions made by a student for an assignment, the system can provide insights into how students approach assignments and the issues they encounter before arriving at a final submission.
- Suggest interventions: Based on observed patterns, the system should make suggestion on how to address these issues.

3.2 Requirements Analysis

This section discusses the requirement analysis by detailing the considerations and resulting requirements.

3.2.1 Considerations

Based on the objective and literature, we identified key considerations for the system design.

- **Explainability**: Trust and transparency in a system are crucial, especially in educational contexts. As noted by Ribeiro et al., "if the users do not trust a model or a prediction, they will not use it" [61]. Moreover, when educational research evidence is framed as AI research, it is deemed less credible compared to when it is framed within educational psychology or neuroscience [62]. Furthermore, an automated system that provides explanations even when it is wrong, can increase the likelihood of adoption of the system opposed to a system that does not provide explanations [63]. Therefore, the analytics should be supported by clear reasoning and references to specific submissions and parts of the submission. More specifically, the system should reference submissions in which it found a specific observation. The user can then take a closer look into the submission to verify the findings. Additionally, the system should provide reasoning within a submission about the observed patterns.
- **Accuracy**: High accuracy is essential for the system's effectiveness. State-of-the-art models should be used to evaluate the potential. We introduce performance metrics in Section 4.3 to measure various aspects of the system.
- **Flexibility**: The system should be flexible enough to adapt to different courses. While fine-tuning the system for a specific course or dataset could significantly enhance performance, it may also limit the system's generalisability to other courses. For example, an LMS that supports various types of (programming) assignments across different courses would benefit from a flexible system. Such a system can be more easily integrated into other courses without requiring extensive tailoring or fine-tuning. This adaptability could reduce the time and effort needed for course-specific modifications and could increase the likelihood of adoption of the system.
- **Actionability**: As described in Chapter 2, a challenge identified in LA research was limited actionability of LA and the lack of interventions that systems can suggest. Therefore, the system should ensure it provides actionable insights.
- **Scalability**: The system must be scalable to handle a large number of students and assignment submissions. It should also be able to analyse programming submissions that are large in size, meaning it contains many lines. Analyses of complete projects are out of scope.
- **Usability**: An intuitive UI is crucial for interacting with the reports. Providing relevant context with the reports is important to inform the user; therefore, the assignment reports should be accompanied with the assignment description and model answer. It should also be easy to navigate to submissions for each assignment, allowing educators to verify the system's findings. Similarly, student reports should provide quick reference to submissions and the submissions should be shown along the assignment description and model answer. Although most learning analytics dashboards have visual elements as described in Chapter 2, our system is text-based and will not contain any visualisations.

3.2.2 Requirements

Based on these considerations, the following requirements were identified:

1. **General**:
   - The system should generate reports on common issues or patterns across the entire course.
   - The system should generate reports for each assignment, highlighting common issues or patterns observed.
   - The system should generate reports for each student, highlighting common issues or patterns observed.
   - The system should generate reports about the process of a student making a submission.
2. **Explainability**:
   - The system should provide detailed reasoning for its observations.
   - It should reference specific submissions where it made an observation.
   - It should reference parts of submissions where observations were made.
   - Users should be easily able to trace back the analytics to the original submissions for verification.
3. **Accuracy**:
   - The system should achieve high accuracy in identifying issues and patterns in student submissions.
4. **Flexibility**:
   - The system should be easily configurable for different courses and assignment types.
5. **Actionability**:
   - The system should suggest interventions on how to address issues that students encounter.
6. **Scalability**:
   - The system should handle a large number of submissions.
   - The system should handle student submissions that are large in size, such as assignments with many lines of code.
7. **Usability**:
   - The UI should display assignment descriptions, model answers, and student submissions alongside the analytics.
   - Navigation should allow users to easily switch between different assignments, students, and submissions.

3.3 System Design

To identify common issues across submissions, one might initially consider putting all submissions into a single prompt and letting an LLM analyse them. However, this approach is not scalable due to the limited context window of LLMs, especially when dealing with possibly hundreds of submissions.

Therefore we consider a different approach by analysing individual submissions first, and then aggregating these analyses. This aligns with the recommendations for prompt engineering in the Related Work chapter (see Section 2.4), where it is advised to break large tasks into smaller steps. Also, the system requirements we laid out also require that the system is explainable. The individual analyses could provide the reasoning for identified issues. Thus, we can tackle the problem of scalability and explainability with this approach.

Given these considerations, our approach involves the following steps:

1. **Individual Analysis**: Each submission is analysed individually to identify issues.

2. **Aggregation of Analyses**: Once all individual submissions have been analysed, these analyses are aggregated and supplied to the LLM to identify common patterns and issues across submissions.

3. **Report Generation**: The LLM generates a report summarising the common issues identified. For explainability, the LLM should reference the specific submissions where these issues were observed. We facilitate this by including the submission ID with each analysis, allowing the LLM to reference the corresponding ID in the report.

4. **Assignment and Student Reports**: To generate reports specific to each assignment and student, we can simply filter the submissions before generating the reports.

5. **Course-Wide Reports**: For course-wide reports, the number of analyses can still be quite large. To manage this, we can again apply the prompt engineering concept of breaking the task into smaller tasks. The assignment reports that are created before can be summarised to produce a course-wide report.

To further enhance the system, we considered the analysis of code revisions as a process. As described in Section 2.1.1, existing efforts leveraging code revisions to produce insights are often lacking in their actionability. To address this, analysing code revisions with LLMs could offer a more comprehensive understanding of students' problem-solving processes by generating natural language descriptions of the code revisions and the progression. This involves adding each revision to the prompt and clearly indicating the distinction between each.

3.4 Data Flow

This section describes the proposed system in more detail and provides a more complete overview. We discuss the context of the prompts, outline the data flow in the system, and describe the technologies used in the system's implementation.

3.4.1 Context in Prompts

Ensuring sufficient context in the prompts is crucial to obtain relevant and high-quality responses. The possible context that can be added includes:

- **Student Submission**: The code that the student submitted.
- **Submission History**: Revisions and changes made to the submission over time.
- **Assignment Description**: Information about the assignment requirements and objectives.
- **Rubric**: Criteria used to evaluate the submission.
- **Model Answer**: The ideal solution or an example solution.
- **Outputs from other tools**: Results from dynamic analysis tools that test functional correctness using predefined inputs and outputs can be included to give the LLM a better impression of the solution. Error messages could also be included such that the LLM can take into account what exactly went wrong.

To enhance the quality and relevance of the suggested interventions, the following context can be considered:

- **Course Setup**: Information about the course structure, such as the presence of weekly lectures, flipped classroom setups, or an overview of the full assignment setup.
- **Learning Goals**: To align the suggestions better with goals of the course.

While all these elements can be included in the prompts, our system design will consider only the data available in the setup of the system evaluation, which will be detailed in detailed in Chapter 4. The data considered in our system design are the submission, revision history, assignment description, and model answer.

3.4.2 Overview

The process of analysing student submissions is divided into the following major steps:

1. **For each student submission:**
    - Retrieve the student's code submission.
    - Retrieve the corresponding assignment description and model answer.
    - Retrieve the code revisions for process analysis.
2. **Analyse each submission:**
    - **Provide General Analysis:**
        - Prompt the LLM with the submission, model answer, and assignment description.
        - Save the generated analysis to the database.
    - **Process Analysis:**

  - Prompt the LLM with the revisions, model answer, and assignment description.
  - Save the process analysis to the database.

3. **Generate Reports:** For both the general and process analysis.
   - **Individual Reports:**
     - Retrieve all analyses of the submissions for a specific student.
     - Prompt the LLM to summarise the analyses and identify common trends and issues.
   - **Assignment Report:**
     - Retrieve all analyses given for a specific assignment.
     - Prompt the LLM to summarise the analyses and identify common trends and issues.
   - **Course-wide Trends:**
     - Retrieve all assignment reports.
     - Prompt the LLM to summarise the assignment reports, identifying common trends and issues.

Figure 3.1 shows a high-level overview of the flow for the general analysis and Figure 3.2 shows the overview for the process analysis. The process analysis is similar to the general analysis, with the difference being the prompt and the input of revisions instead of the final submission.

The primary stage utilises the LLM to conduct an analysis of code submissions, focusing on two aspects: general feedback and process analysis. For the general analysis, the LLM conducts an analysis of a submission with respect to the assignment description and model answers. This analysis is open-ended, allowing the LLM to identify and highlight any relevant issues or noteworthy points without being specifically instructed on what to look for. The process analysis considers the revision history of a submission as explained earlier.

Based on the individual submission analyses, reports are generated by the LLM that provides insights at multiple levels:

- **Individual Report:** Taking into consideration all the analysis generated for a specific student.
- **Assignment Reports:** Focuses on submissions made by all students for one assignment, identifying common issues and patterns.
- **Course-wide Report:** Gives an overview of trends and patterns throughout the course, by analysing all the submissions that were made for each assignment.

3.4.3 Technologies Used

Initially, LangChain [64] was used for two main reasons. Firstly, LangChain offers the flexibility to use various LLMs, making it easy to switch between different models. Secondly, it provides functionality to define and extract structured outputs from the LLM. This is useful, for instance, for separately parsing the grade and feedback from the LLM's response.

Over time, we learned to instruct the model to provide JSON responses consistently, enabling us to parse the responses directly. Furthermore, as the reasoning

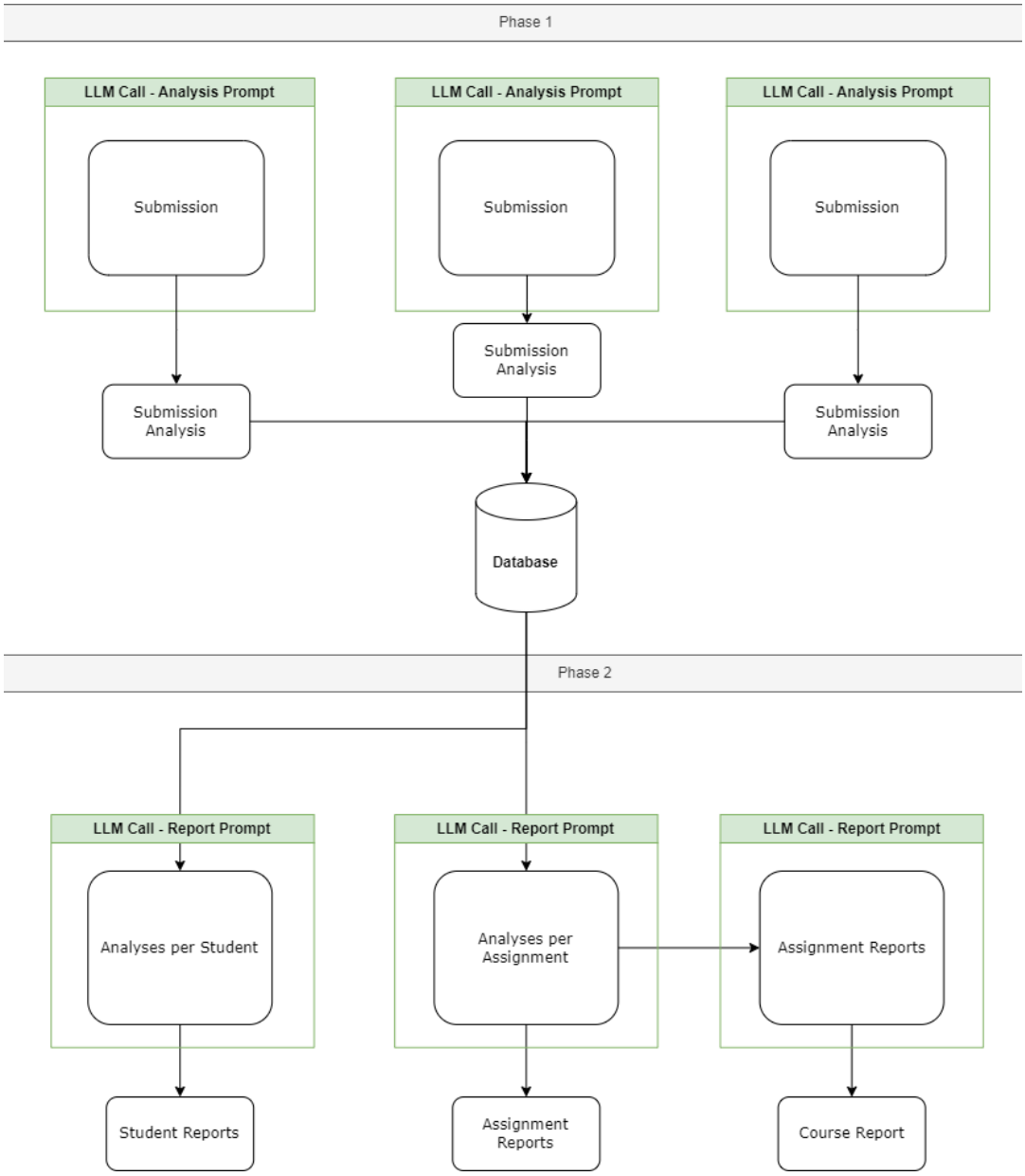

FIGURE 3.1: High-level overview of the analysis of submissions and the generation of reports for individuals, assignments, and the entire course.

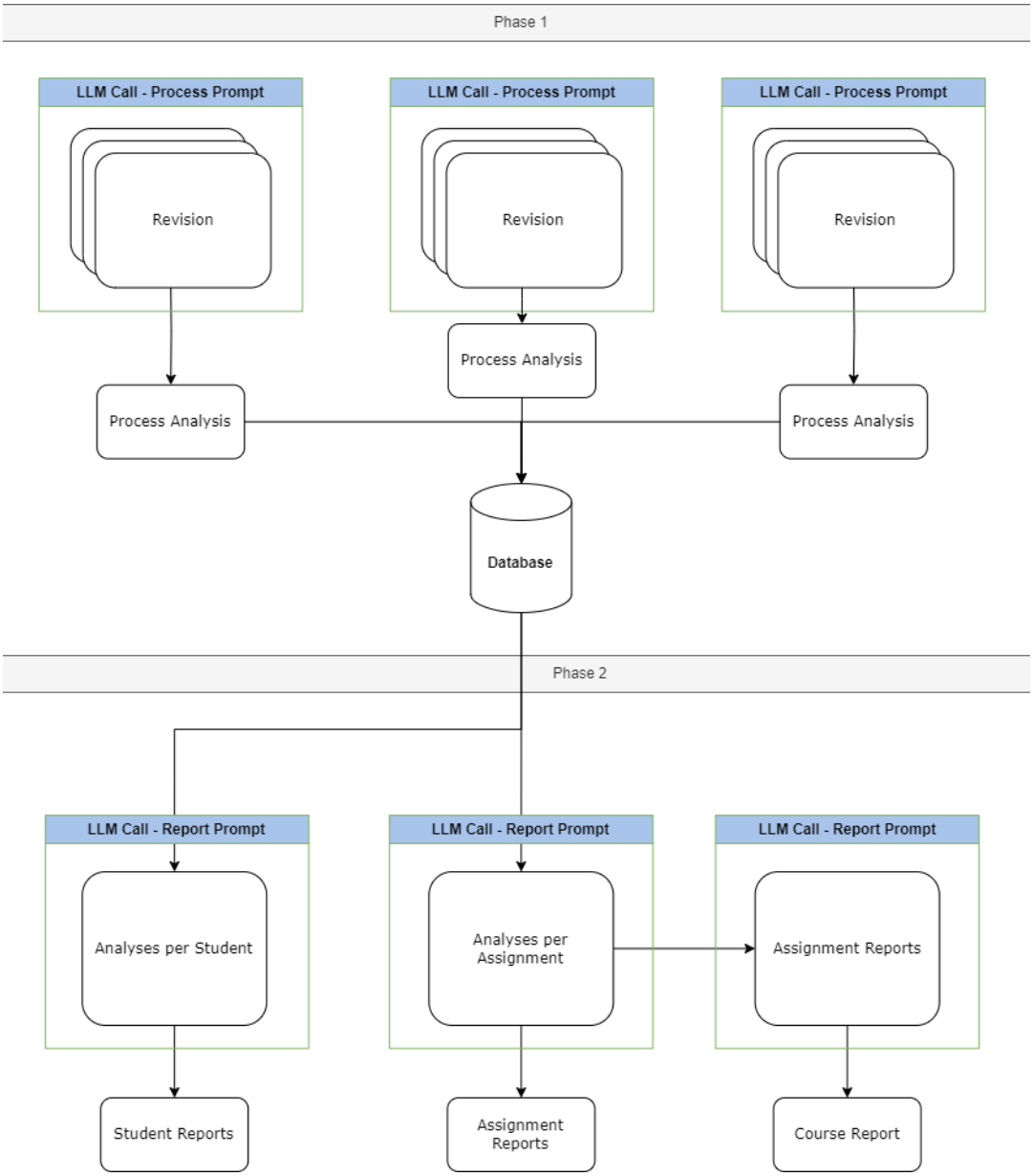

FIGURE 3.2: High-level overview of the process analysis of submissions through the revision history, and the generation of reports for individuals, assignments, and the entire course.

capabilities of the LLM are essential to analyse submissions effectively, we decided to settle for one model and use the state-of-the-art model GPT-4o [65]. Additionally, GPT-4o supports a 128k context window, offering better scalability compared to GPT-3.5, which has a context window of 16k. Consequently, we decided to simplify the approach by discontinuing with LangChain and using OpenAI's API directly.

Python was used to interact with the API and an SQLite database was made to store the analyses and reports.

3.5 Prompts

In this study, the design of the prompts is important in instructing the LLM to analyse student code submissions effectively. The prompts are designed using prompt engineering techniques to ensure that the responses provided by the LLM are relevant and of high-quality. This section highlights the three main prompts. Each prompt contains placeholders that add more context.

3.5.1 Analysis Prompt

This prompt instructs the LLM to give a general analysis of an individual assignment submission. The prompt strategies used are 'adopting a persona' and the use of clear delimiters with XML tags.

In addition to giving a textual description of the analysis, the prompt also includes instructions to classify the submission. This was necessary as part of the system evaluation discussed in Chapter 4. Section 4.2.3 elaborates on the prompt design in the context of the dataset. For the system itself in practice, the classification is unnecessary and the 'analysis' is what is essential to store in the database for the next step to generate analytics.

```
You are a teacher grading a student's submission for an SQL assignment.
You are given the assignment description, the student's SQL submission
    statement, and a model answer.
Perform an analysis of the student's submission and classify the
    submission in one or more of the following categories: '
    non_interpretable', 'partially_correct', 'correct', 'cheating'.
cheating: (4) when you suspect cheating or their approach bypasses the
    intended logic and problem-solving process. For example, cheating
    in an SQL assignment can occur when the student hardcodes specific
    values directly into the query, even though these values were not
    explicitly mentioned in the assignment.
non_interpretable: the statement is non-executable, containing syntax
    errors.
correct: the execution result of the SQL statement is the same as the
    expected result.
partially correct: the execution result of statement is different from
    the expected result, but the SQL statement can be interpreted,
    meaning it is executable.

Follow this Pydantic model to provide your analysis:
    class Analysis(BaseModel):
        analysis: str = Field(..., description="Step-by-step analysis of
            the submission.")
```

```
        category: List[ActionChoices] = Field(..., description="Category
            of the submission")
    class ActionChoices(int):
        non_interpretable = 1
        partially_correct = 2
        correct = 3
        cheating = 4

<assignment_description>
{assignment_description}
</assignment_description>

<student_submission>
{student_submission}
</student_submission>

<model_answer>
{model_answer}
</model_answer>

Please output valid JSON.
```

The Pydantic model part of the prompt in combination with the instruction to output valid JSON was added to parse the output conveniently. The LLM will then respond with a JSON object such as the following:

```
{
  "analysis": "The student submission is partially correct. The
     solution includes [...]",
  "category": [2]
}
```

We can then easily parse this and save the results in our database.

3.5.2 Process Analysis Prompt

This prompt focuses on the process of the student during the making of the assignment, using the code revisions. Here, we make use of chain-of-thought prompting by breaking the task into steps. The instructions also mention that the LLM should not repeat the code revisions in its response to remove redundancy. These revisions can be displayed in the UI alongside the report.

```
You are a teacher analyzing a student's work on an assignment.
You are given the assignment description, model solution, and sequence
    of submissions made by the student.

<assignment_description>

{assignment_description}

</assignment_description>

<model_solution>
```

```
{model_answer}

</model_solution>

It is important to evaluate the revisions and overall development
    process based on the sequence of submissions.
To do this systematically, describe each of the submissions made by the
     student with respect to the previous one to describe the process.
Please consider the submissions below and ensure the submissions are
    relevant to the assignment description. DO NOT display the
    submissions again in your response.

<submission1>

{submission1}

</submission1>

<submission2>

{submission2}

</submission2>

{...}

Guidelines for Analysis:
- Initial Review: Identify the primary issues with each submission
    compared to the model solution.
- Progress Evaluation: Note improvements or regressions between
    submissions.
- Final Assessment: Provide an overall evaluation of the student's
    progress and understanding of the assignment.
Now, analyze the student's submissions accordingly, and do not write
    out the full submissions in your reply.
```

3.5.3 Report Generation Prompt

The following prompt structure was used to generate the assignment, student, and course reports. It aggregates the individual submission analyses by summarizing the common challenges, referencing specific submissions, and suggesting interventions to address the issues.

To add the individual analyses into the prompt, the analyses are encoded into a dictionary where each submission ID serves as the key and the corresponding analysis as the value. The LLM is then instructed to reference these submission IDs in its summary to highlight specific observations.

```
You are given a dictionary containing an analysis for student
    submissions. The dictionary has the submission ID as the key and
    the analysis as the value.
```

```
{analysis_dictionary}

Identify and summarize the most prevalent issues, challenges, and
    submission characteristics highlighted in the analyses.
Give a list of submission IDs where the issues were identified. And
    give concrete examples.
Also, highlight anomalies or outliers in the analyses that should be
    mentioned to the teacher.
Based on these common themes, suggest targeted interventions or
    strategies that teachers can implement to effectively address and
    resolve these issues.
```

3.6 User Interface

A user interface (UI) was developed to help educators in the user study in Chapter 5 easily use and understand the learning analytics generated by the LLMs. The UI was developed using Streamlit [66], an open-source Python framework that provided a simple connection with our data. Streamlit was chosen for its simplicity in creating an interactive and visually appealing web application. Key features of the UI include:

1. **Assignment, Student, and Submission Navigation**: Educators can quickly switch between different assignments, students, and submissions using the sidebar menu that lists all of them.

2. **Submission Viewing**: Educators can see the analysis made for each submissions next to the submitted code and model answers, allowing for easy comparison and tracing.

It was important to add navigation to submissions so educators can dive deeper into the submission to observe patterns described by the reports. Figure 3.3 shows the UI used in the study for showing assignment reports. Appendix A shows examples of the other pages of the UI.

3.7 Discussion

As discussed earlier, LLMs can suffer from hallucinations, presenting information as true when it is not. One example of this issue is when the LLM is instructed to provide statistics. Originally, the prompt contained instructions to give a percentage of how often an issue was present. It quickly became apparent that these percentages were not accurate.

To provide accurate statistics, we considered having the LLM classify issues using specific coded tags. For example, when a submission is inefficient, the LLM could respond with a tag such as 'inefficient'. We could then count all the submissions with this tag to provide statistics. However, this approach would require predefined tags. The problem with predefined tags is that the LLM might only look for those specific issues, potentially overlooking other issues present in the submissions that were not anticipated.

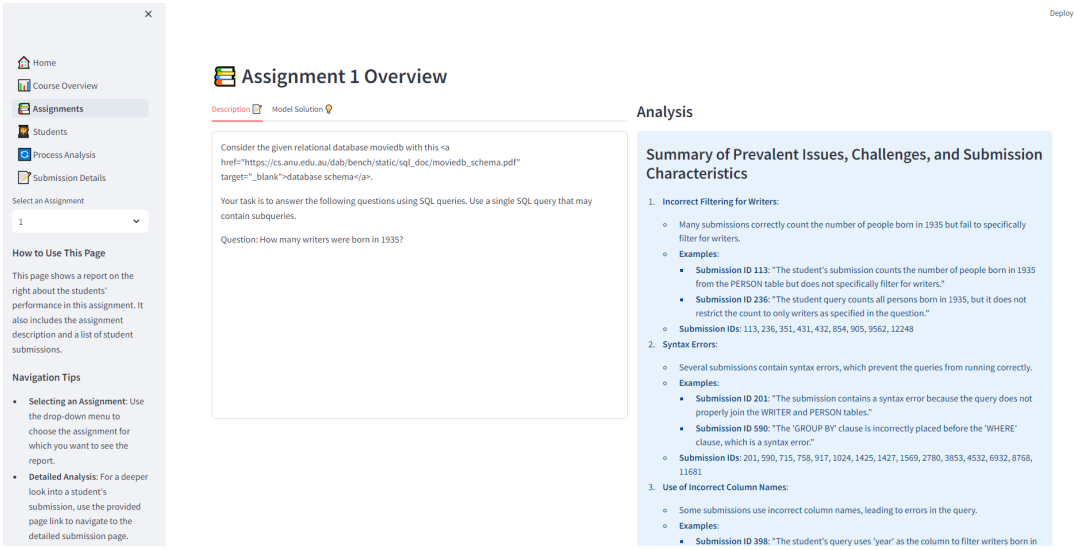

FIGURE 3.3: User Interface Overview for the Assignment Report

To address this, we considered allowing the LLM to define tags on the spot without predefining them. The problem with this approach is the inconsistency in tagging. For instance, the LLM might tag one submission as 'wrong_variable' and another as 'variable_wrong', making it difficult to accurately count them as the same tag programmatically.

While statistics could enhance the reports, we decided to continue with the approach explained in this section. Whether there is a need for exact statistics or if a list of a few examples of submissions is sufficient will become clear in the user study.

This also leads us to a limitation of the requirements analysis, which is the absence of stakeholder input at this phase. Consulting relevant stakeholders could have possibly clarified the necessity for exact statistics and introduced other considerations or requirements.

Additionally, integrating this system into existing LMS would be more practical than using it as a standalone application. Integrating it in an existing system was not feasible. Instead we present a UI to provide an idea of the user interaction. The user study with this UI could make clear which design considerations are crucial for implementation within existing LMS.

Chapter 4

System Evaluation

In this chapter, we evaluate the system. Section 4.1 states the objective of the evaluation. In Section 4.2, we describe the dataset used for evaluation, including the pre-processing steps taken. Next, Section 4.3 outlines the metrics used to evaluate various aspects of the system. Section 4.4 presents the results and analysis, discussing the system's performance based on the defined metrics and interpreting the generated reports. Finally, Section 4.5 discusses the findings as a whole, offering insights into the implications and potential improvements for the system.

4.1 Objective

In this section, we break down the ideal evaluation process and how we actually evaluated it given our constraints.

First, the accuracy of the analysis made by the LLM for a single submission is crucial. Ideally, subject matter experts with experience in grading the assignment would rate the analysis. This process would require extensive manual work, as it involves detailed assessment by these experts.

For the generated reports, it is essential that the mentioned issues are accurate and that the submissions are correctly referenced. Ideally, this involves a dataset where the issues in the submissions are labelled by subject matter experts. However, labelling such a dataset is laborious and time-consuming.

Besides the accuracy of the system, it is also important to evaluate the system's usefulness for educators. Educators should assess whether the system can actually support their work effectively in a course setting. This involves user evaluations with the system, possibly over a long period.

However, the ideal situation for this study was not feasible due to limited availability of appropriate datasets that could also be evaluated by relevant subject matter experts. Additionally, the manual labelling that would be required posed limitations. Despite efforts to facilitate this methodology, it ultimately proved to be impractical.

As a result, we adopted the following approach:

- We used a publicly available dataset with individual submissions classified into categories. The accuracy of these classifications in combination with the LLM's analysis leading to the classification provides an indication of the accuracy of the analysis, though it is not as comprehensive as an expert evaluation.
- For the reports, we manually verified whether the mentioned issues were actually present in the referenced submissions.
- For the user evaluation, we refer to Chapter 5.

4.2 Dataset

In this section, we provide an overview of the dataset used in this thesis. The dataset includes student submissions to SQL exercises from a university course. The following subsections detail the contents of the dataset, the rationale for using this dataset, and pre-processing steps taken.

4.2.1 Dataset Description

The dataset used in this research came from a study conducted in an undergraduate Relational Database course at the Australian National University [31]. The data was collected online over a three-week period beginning on August 10th, 2018, involving 393 students. These students were tasked with completing 15 SQL exercises on an online assessment platform that provided an SQLite environment in their browsers. The platform allowed students to submit and execute their SQL statements, providing feedback when their submissions matched the expected results.

They recorded a total of 12,899 SQL submissions and based on their execution results they were first classified into three categories. After that, the submissions classified as 'correct' were manually analysed to observe if it could be categorised into a 4th category 'cheating':

- **Non-interpretable**: The statement is non-executable.
- **Partially Correct**: The execution result differs from the expected result, but it is executable.
- **Correct**: The execution result matches the expected result.
- **Cheating**: The submission is an attempt to deceive the system by hardcoding values to produce correct results. For example, selecting a specific ID that matches the condition of the expected result.

The dataset also contained descriptions of the exercises and model solutions. The following is an example of an exercise description:

- *Consider the given relational database moviedb with this database schema [URL]. Your task is to answer the following questions using SQL queries. Use a single SQL query that may contain subqueries. Question: How many persons have never directed any movies in this database? List the total number of such persons.*

4.2.2 Rationale for Dataset Selection

This dataset was selected for two main reasons. The dataset contained programming submissions from multiple students across multiple exercises, enabling us to create aggregate reports at both the student and exercise levels. This facilitates the identification of common issues.

Moreover, its labelling of SQL submissions into four categories was important. This allowed us to evaluate an LLM's performance in classifying a submission, serving as an indication of the accuracy in analysing a submission.

4.2.3 Data Exploration and Pre-Processing

Table 4.1 shows the number of submissions that fall into each of the four categories along with their respective percentages.

Category	Number of Submissions	Percentage
Non-interpretable	2134	16.5%
Partially Correct	3630	28.1%
Correct	7122	55.2%
Cheating	13	0.1%

TABLE 4.1: Number of submissions in each category with percentages

Given the large size of the dataset and the manual analysis required by us, we took the following steps to reduce the size of the dataset:

1. **Including 'cheating' Submissions**: We first selected the exercises where all four categories were present. Since 'cheating' submissions were rare, only a few exercises were selected.
2. **Difficulty Levels**: The exercises had different difficulty levels. Two additional exercises were selected to include more difficult exercises.
3. **Sampling**: The dataset was reduced to 500 submissions, maintaining a similar distribution of categories compared to the full dataset and ensuring that all 'cheating' submissions were included.

Exercises 1, 3, 5, and 7 were identified as exercises containing all four categories. Additionally, exercises 9 and 14 were included because they were labelled with difficulty 4 and 5, whereas exercises 1, 3, 5, and 7 only had difficulties 2 and 3.

Categories in Selected Exercises

Table 4.2 shows the number of submissions that fall into each category for the selected exercises (1, 3, 5, 7, 9, and 14) along with their respective percentages.

Category	Number of Submissions	Percentage
Non-interpretable	896	17.1%
Partially Correct	1474	28.1%
Correct	2861	54.6%
Cheating	13	0.2%

TABLE 4.2: Number of submissions in each category for selected exercises with percentages

Sampled Subset

Due to the high number of submissions, we reduced the dataset to 500 submissions. Table 4.3 shows the distribution of these submissions across the four categories with percentages.

Category	Number of Submissions	Percentage
Non-interpretable	92	18.4%
Partially Correct	137	27.4%
Correct	258	51.6%
Cheating	13	2.6%

TABLE 4.3: Number of submissions in each category for the reduced dataset with percentages

Table 4.4 shows the number of submissions in each exercise within the reduced dataset.

Exercise ID	1	3	5	7	9	14
Number of Submissions	132	69	79	68	84	68

TABLE 4.4: Number of submissions in each exercise

Data for Process Analysis

Initially, our primary focus was on the general analysis and generation of reports. After completing this phase, we realised that the dataset could contain multiple submissions from a single student for a single exercise, which can be seen as a process when ordering these submissions by time. This was not taken into account in the general analysis, and therefore data exploration was done again focusing on the process.

To begin, we plotted a histogram showing the distribution of the total number of submissions each student made for a single exercise, aggregated across all exercises. Figure 4.1 illustrates this histogram.

From the histogram, we saw that most students completed an exercise in 1 or 2 submissions. For the purpose of process analysis, we excluded 'processes' consisting of only one submission since this would not make sense to evaluate as a process.

Moreover, we aimed to balance the dataset. We did not want to include only processes with, for example, just two submissions but also wanted to include processes with more than ten submissions. This was crucial to determine if the system could effectively summarise a process with varying submission counts.

Ultimately, we reduced our dataset to a subset that included processes with each of the number of submissions between 2 and 20 only once. This subset allowed us to perform a manual analysis on a manageable number of submissions.

Prompt Design

Initial tests indicated that the LLM could detect hardcoded values but classified the statements as non-interpretable if it also contained syntax errors. To address this, the LLM was allowed to predict multiple classes for each submission. A heuristic approach was then applied to refine these predictions: if a statement was classified as both non-interpretable and cheating, it was classified as cheating. If classified as non-interpretable and partially correct, the non-interpretable classification was chosen.

Furthermore, the task description of each exercise in the dataset references a database schema in an external PDF. However, we decided not to include this schema

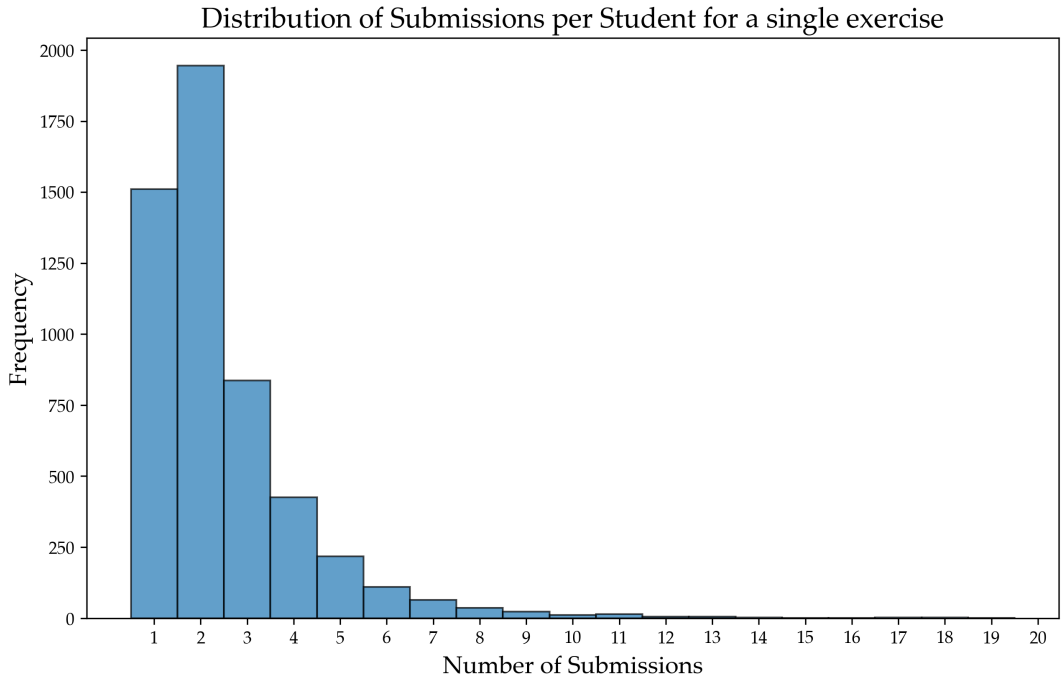

FIGURE 4.1: Distribution of Submissions per Student for a single exercise.

in the prompt. Instead, the model solution added to the prompt should provide sufficient context about the database schema to evaluate the submission effectively.

For example, consider the following model solution:

```
SELECT p.id
FROM person p
WHERE p.year_born = (SELECT MAX(year_born)
    FROM person
    WHERE year_born < (SELECT MAX(year_born)
        FROM person));
```

This solution contains the necessary information about the database tables ('person') and columns ('id', 'year_born').

4.3 Evaluation Methodology

The evaluation methodology for the system first involves assessing the performance of the underlying LLM in classifying the student submissions in the dataset. To evaluate the reports, the precision of the referenced submissions in the report is measured.

4.3.1 Performance Metrics

To assess the performance of the LLM in classifying the submissions into the four categories, several standard evaluation metrics are employed. These metrics are Accuracy, Precision, Recall, F1-score, and the balanced accuracy:

Accuracy: This metric measures the overall correctness of the model's classifications. It is calculated as the ratio of correctly classified instances to the total number

of instances.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{All Predictions}}$$

Precision: Precision measures the accuracy of the model's positive predictions. It is important in scenarios where the cost of a false positive is high. It is defined as the ratio of true positive predictions to the sum of true positive and false positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall: Recall measures the model's ability to identify all relevant instances. It is important in scenarios where missing a positive instance has significant consequences. It is calculated as the ratio of true positive predictions to the sum of true positive and false negative predictions.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-Score: The F1-score is the harmonic mean of precision and recall and balances both aspects. It is computed as follows:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Balanced Accuracy: This metric is useful in multiclass classification when dealing with unbalanced class distributions. It is essentially the average of each class' Recall scores.

$$\text{Balanced Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

where N is the number of classes.

These metrics are important for understanding the effectiveness of the LLM in classifying a submission. Calculating the Precision, Recall, and F1-score for each of the categories helps us identify areas where the model performs well and areas where the model is not as good.

4.3.2 Precision for Reports

In addition to evaluating the classification performance of the LLM, it is important to assess the generated reports. The reports generated by the system highlight common issues and patterns. Ideally, the dataset would be labelled with all issues and observed patterns in each submissions, but the dataset used in this thesis does not contain that data. It would also require a lot of time and effort to manually label all possible issues that can be observed in each submission.

What we instead can measure is the precision of the referenced submissions in the report. Each report generated by the LLM contains common issues or patterns that were observed, and the LLM is asked to also reference specific submissions where it observed these patterns. To verify if these references are accurate and that a reference is not another example of 'hallucination' by the LLM, we manually check the referenced submissions. This involves checking for each referenced submission if it actually contains the observed issue mentioned in the report.

Based on the number of correct retrieved references and the number of total retrieved references, we can calculate the precision of the generated reports:

$$\text{Report Precision} = \frac{\text{Number of Correct References}}{\text{Total Number of References Retrieved}}$$

4.4 Results and Analysis

In this section, we present and analyse the results of the evaluation. We start by evaluating the performance metrics of the LLM, including a comparison with other models. Next, we take a look at the confusion matrices, providing insights on misclassifications. The generated reports will then be described along with the metric we used to evaluate reports.

4.4.1 Performance Metrics

We evaluated the performance on the dataset with both GPT3.5 and GPT-4o models. The accuracy was 42.6% with the GPT-3.5 model and 64.4% with the GPT-4o model. Table 4.5 provides a summary of key performance metrics of the GPT-4o model, including precision, recall, and F1-score for each category, alongside the accuracy.

Category	**Precision**	**Recall**	**F1-Score**
Non-interpretable	0.77	0.88	0.82
Partially correct	0.45	0.75	0.56
Correct	0.84	0.51	0.64
Cheating	0.86	0.46	0.60
Overall Accuracy: 0.64, **Balanced Accuracy:** 0.65			

TABLE 4.5: Performance Metrics Summary of the GPT-4o Model.

The performance metrics indicate that 'non-interpretable' submissions were identified quite well, achieving an F1-score of 0.82. However, the F1-scores for other categories were lower. For 'partially correct' submissions, the precision was particularly low. In contrast, the precision for 'correct' and 'cheating' submissions was quite high, indicating that when the model predicted a submission as correct or cheating, it was usually accurate. However, the recall for these categories was lower, indicating that the LLM missed a significant number of actual 'correct' and 'cheating' submissions. This suggests that while the LLM is good at identifying true positives in these categories, it struggles to capture all relevant instances, leading to more false negatives.

Comparison to BERT

Two studies that were mentioned earlier measured the performance of a convolutional neural network (CNN) [32] and BERT [29] on the same dataset. Table 4.6 shows the accuracy and balanced accuracy. Table 4.7 shows a comparison of the performance in other metrics, with BERT outperforming both models.

GPT-4o, with an accuracy of 64%, shows similar overall accuracy to the baseline CNN, which had an accuracy of 66%. However, GPT-4o significantly outperforms the baseline in terms of balanced accuracy, achieving 65% compared to CNN's 37%.

	CNN	BERT	GPT
Accuracy	0.66	**0.90**	0.64
Balanced Accuracy	0.37	**0.93**	0.65

TABLE 4.6: Performance Metrics Summary of Different Models.

Class Being Evaluated	Evaluation Metrics											
	Precision			Recall			F1-score			Support		
	CNN	BERT	GPT	CNN	BERT	GPT	CNN	BERT	GPT	CNN	BERT	GPT
Non Interpretable	0.26	0.74	**0.77**	0.09	0.85	**0.88**	0.13	0.79	**0.82**	57	20	92
Partially Correct	0.51	**0.85**	0.45	0.58	**0.90**	0.75	0.54	**0.87**	0.56	219	356	137
Correct	0.78	**0.95**	0.84	0.80	**0.91**	0.51	0.79	**0.93**	0.64	393	632	258
Cheating	0.0	**1.0**	0.86	0.0	**1.0**	0.46	0.0	**1.0**	0.60	6	1	13

TABLE 4.7: Classification Performance Analysis with the baseline and BERT.

This indicates that GPT-4o can handle minority classes much better than the baseline. It shows that the model is not biased towards the majority classes ('correct' and 'partially correct') and that it can identify instances from the minority classes more accurately.

BERT consistently outperforms CNN and GPT-4o in most classes, achieving the highest scores in almost all evaluation metrics. The 'non-interpretable' class presents an exception, where GPT-4o outperforms BERT.

Furthermore, the support for the 'cheating' class is notably low for BERT, with only one occurrence. This means that BERT successfully identified the one instance of 'cheating', resulting in perfect scores (1.0) for precision, recall, and F1-score in this class. However, these high scores are based on a single instance, which limits the significance and makes it harder to compare the metrics in this class.

4.4.2 Confusion Matrix

A confusion matrix helps us compare the predicted categories with the actual categories. The confusion matrix for the GPT-4o model is detailed in Table 4.8.

From the confusion matrix, we can see that the majority of misclassifications (115) were when the actual category was 'correct' while the LLM predicted 'partially correct'.

	Predicted Category			
Actual Category	**Non-interpretable**	**Partially correct**	**Correct**	**Cheating**
Non-interpretable	**81**	11	0	0
Partially correct	8	**103**	25	1
Correct	11	115	**132**	0
Cheating	5	2	0	**6**

TABLE 4.8: Confusion Matrix for the GPT-4o Model Performance on the SQL Submission Classification.

To further understand the performance, we created confusion matrices for each of the six exercises in the dataset. These individual matrices could help identify if there were exercises where the LLM struggled more than others.

	Predicted Category			
Actual Category	**Non-interpretable**	**Partially correct**	**Correct**	**Cheating**
Non-interpretable	**18**	2	0	0
Partially correct	1	**30**	24	0
Correct	0	15	**40**	0
Cheating	2	0	0	**0**

TABLE 4.9: Confusion Matrix for Exercise 1 with difficulty 2. (n=132)

	Predicted Category			
Actual Category	**Non-interpretable**	**Partially correct**	**Correct**	**Cheating**
Non-interpretable	**16**	0	0	0
Partially correct	0	**11**	0	0
Correct	2	8	**24**	0
Cheating	3	1	0	**4**

TABLE 4.10: Confusion Matrix for Exercise 3 with difficulty 3. (n=69)

	Predicted Category			
Actual Category	**Non-interpretable**	**Partially correct**	**Correct**	**Cheating**
Non-interpretable	**8**	2	0	0
Partially correct	3	**6**	0	1
Correct	4	44	**9**	0
Cheating	0	0	0	**2**

TABLE 4.11: Confusion Matrix for Exercise 5 with difficulty 2. (n=79)

	Predicted Category			
Actual Category	**Non-interpretable**	**Partially correct**	**Correct**	**Cheating**
Non-interpretable	**11**	1	0	0
Partially correct	2	**14**	0	0
Correct	2	8	**29**	0
Cheating	0	1	0	**0**

TABLE 4.12: Confusion Matrix for Exercise 7 with difficulty 3. (n=68)

	Predicted Category			
Actual Category	**Non-interpretable**	**Partially correct**	**Correct**	**Cheating**
Non-interpretable	**9**	4	0	0
Partially correct	1	**27**	0	0
Correct	1	33	**9**	0
Cheating	0	0	0	**0**

TABLE 4.13: Confusion Matrix for Exercise 9, with difficulty 4. (n=84)

	Predicted Category			
Actual Category	**Non-interpretable**	**Partially correct**	**Correct**	**Cheating**
Non-interpretable	**19**	2	0	0
Partially correct	1	**15**	1	0
Correct	2	7	**21**	0
Cheating	0	0	0	**0**

TABLE 4.14: Confusion Matrix for Exercise 14, with difficulty 5. (n=68)

From the confusion matrices and the calculated accuracy for each exercise in Table 4.15, we can observe that two exercises in particular had notably low accuracy. Exercise 5 and 9 demonstrated significantly lower performance compared to the other exercises, with an accuracy of 0.32 and 0.54 respectively. Therefore, we took a closer look into the reasoning of the classifications for these two exercises to understand what led to the misclassifications.

Exercise	**Accuracy**
Exercise 1	0.67
Exercise 3	0.80
Exercise 5	0.32
Exercise 7	0.79
Exercise 9	0.54
Exercise 14	0.81

TABLE 4.15: Accuracy for Each Exercise.

4.4.3 Highlighting Misclassifications

Before arriving at the predicted categories, the model was asked to provide a step-by-step analysis of the submission and provide reasoning for the classification. We inspected the reasoning for several misclassified submissions to provide a more nuanced assessment of the LLM's performance.

4.4.3.1 Partially Correct

Most wrongly classified instances were between 'partially correct' and 'correct'. The confusion matrices per exercises showed that 2 exercises in particular had a lot of disagreement. These are Exercise 5 in Table 4.11 and Exercise 9 in Table 4.13. The description of these exercises are as follows:

- **Exercise 5:** *Your task is to answer the following questions using SQL queries. For each question, your answer must be a single SQL query that may contain subqueries. Question: Which movies were written by Kevin Williamson? List the titles and production years of these movies.*
- **Exercise 9:** *Your task is to answer the following questions using SQL queries. For each question, your answer must be a single SQL query that may contain subqueries. Question: How many writer awards have been given to Woody Allen between 1991 and 1995 (inclusive)? List the number of the awards.*

Our analysis of the misclassified submissions revealed some patterns. The LLM frequently marked submissions as 'partially correct' due to case sensitivity issues with the 'LIKE' operator, even though the database system used in the exercise was case-insensitive [67]. The LLM did not have this context, so more context about the SQL environment could have prevented these misclassifications. Similarly, the use of single quotes versus double quotes also depends on the environment and led to misclassifications that could have been avoided with more contextual information about the SQL environment.

Generally, it seemed that the analyses focused too much on comparing student submissions with the model answer. While student submissions often produced the correct output, the LLM highlighted that it could be improved compared to the model answer. For instance, the model solution explicitly specified join conditions while some submissions did it with a natural join, resulting in 'partially correct' classifications when submissions did not follow this approach. In some cases, the LLM acknowledged that the output was correct but still classified the submission as partially correct due to differences from the model answer. However, the stricter analysis by the LLM could also be seen as an advantage, since it can identify improvements in correct submissions. The areas for improvement can inform educators on how the students could improve, possibly leading to better learning outcomes.

There were also some cases where the submission produced the correct output due to specific entries in the database, but with more entries it would not have produced the correct output. For instance, in the exercise about finding the number of awards won by Woody Allen, a submission matched only on the first name 'Woody', which would find anyone with the first name 'Woody'. Because there were no other Woody's in the database, this submission produced the correct output. However, if there were another person named Woody, the submission would not produce the correct output. The LLM misclassified these submissions as 'partially correct' but was actually correct in its analysis. This highlights the advantage of the LLM over automated tests, as the LLM could recognise the underlying issue instead of purely matching the outputs.

Overall, the majority of misclassifications can be attributed to missing context about the SQL environment or to an excessive focus on matching the model answer. To mitigate these issues, the prompts should include the required context and emphasise that multiple correct solutions may exist.

4.4.3.2 Cheating Submissions

To understand why the model misclassified several 'cheating' submissions, we took a closer look at these predictions and submissions. Our observations indicate that some submissions in the original dataset were labelled arbitrarily.

Arbitrary Classification

During our review, we identified instances where the classification of submissions as 'cheating' seemed arbitrary. This was evident in Exercise 1, where students were asked to create an SQL statement for the question: "How many writers were born in 1935?". Two submissions in the labelled dataset were marked as 'cheating', despite the content suggesting otherwise. The submission in both cases was:

```
select count(distinct 1935) from writer;
```

The LLM in this case marked it as 'non-interpretable' and gave the following reasoning:

> The student's submission contains a syntax error and does not correctly address the problem statement. The query 'select count(distinct 1935) from writer;' is syntactically incorrect because it is counting the distinct value '1935' rather than counting the number of writers born in 1935. The correct approach should involve joining or subquerying the 'person' and 'writer' tables to filter writers born in 1935.

In this case, it is arbitrary to say that the student attempted to cheat in this exercise as it seems it was a genuine attempt by the student.

Another arbitrary 'cheating' classification in the original dataset was for three submissions by the same student for Exercise 3, in which the question was: "*Assume persons who were born in the same year are the same age and there is only one youngest person (with no ties/draws) in this database, who is/are the second youngest person(s) in the database? List the id(s) of the person(s).*".

The three submissions by the student were flagged as 'cheating', presumably because the student used 'DELETE' in their query to delete rows in the database. The LLM marked these submissions as 'non-interpretable'.

```
DELETE FROM person p
WHERE p.year_born=(SELECT MAX(p.year_born) FROM p)
SELECT id.p
FROM person p
WHERE p.year_born=(SELECT MAX(p.year_of_born)FROM p);
```

One of the analyses for these submissions was as follows:

> The student's submission contains several issues. Firstly, the DELETE statement is unnecessary and incorrect for this query as it modifies the database, which is not required. The intended task is to retrieve data, not delete it. Secondly, there are syntax errors in the SQL statement. The subquery references 'p' without aliasing it correctly and there is a typo in 'year_of_born' instead of 'year_born'. The correct approach would involve using subqueries to find the second highest year of birth without deleting any data.

In this case it is also arbitrary to say the student attempted to cheat, and one can argue it was a genuine attempt to solve the exercise.

Remaining Cases

6 out of the 13 cases of 'cheating' were correctly identified by the LLM, and 5 of the misclassified cases could be attributed to arbitrary classification as observed in the subsection before. Upon observing the remaining two cases, we see that the LLM did mention in the analysis that the students hardcoded specific values without any indication of those in the exercise description. However, the LLM then failed to classify it as 'cheating' and marked it as 'partially correct' as the submission was deemed interpretable but not fully correct:

> The student's submission attempts to group persons by their birth year and then filter for those born in 1988. However, this approach is incorrect for several reasons. First, it uses a hardcoded year (1988) instead of dynamically determining the second youngest person's birth year. Second, the use of GROUP BY and HAVING clauses is inappropriate for

this problem, as these clauses are used for aggregation and filtering aggregated results, not for finding specific rows based on dynamic conditions. The correct approach involves nested subqueries to first determine the maximum birth year, then find the second maximum birth year, and finally select persons born in that year.

Overall, the explanations by the LLM give a different perspective to the classifications. They showed that there is room for improvement to increase the accuracy by enhancing the prompts, but also that LLMs can provide explanations that can be more valuable than a classification.

4.4.4 Generated Reports

In this subsection, we discuss the reports generated by the LLM based on the analysis of student submissions. The reports highlight prevalent issues, challenges, common patterns, and anomalies identified in the submissions. The reports mention 4 to 6 common issues, 1 to 3 anomalies, and also 4 to 6 suggested interventions. The reports also reference the IDs of specific submissions as instructed. Figure 4.2 shows a snippet of the report for Exercise 1, shortened for simplicity.

The report reveals numerous observations that would have required a lot of manual effort to detect. For instance, common pitfalls were identified such as the use of 'NATURAL JOIN', which should generally be avoided. Joining tables was also an issue as some submissions counted all individuals born in 1935 without filtering specifically for writers.

Detecting such issues manually in thousands of submissions would be very time-consuming for an educator unless they specifically searched for these common mistakes. Moreover, many automated systems typically do not catch these errors unless they are explicitly programmed to look for it case-by-case.

The LLM-generated reports not only highlight issues but also provide suggestions to address them. For example, the reports suggest conducting lessons on the proper use of SQL joins and avoiding NATURAL JOIN, which could lead to unintended results.

The reports for other exercises highlighted similar problems as this report, indicating uses of (NATURAL) JOINs and overly complex queries.

Manual Verification

The LLM was instructed to give specific examples of submissions that illustrated the observed issues. To assess if the referenced submissions in the reports actually contained the mentioned observations, we manually checked each referenced submission in the report in Figure 4.2. Table 4.16 presents the results of this verification and the calculated precision for each observation of the LLM. The precision was quite low, with an overall precision of 0.48.

Summary of Prevalent Issues, Challenges, and Submission Characteristics

1. Incorrect Table Joins or Missing Joins:

- **Issue:** Many students failed to correctly join the `writer` and `person` tables or missed the join entirely.
- **Examples:**
  - Submission 113: Counts people born in 1935 without filtering for writers.
  - Submission 236: Counts all persons born in 1935, not just writers.
  - Submission 431: Selects all columns from the `Person` table where `year_born` is 1935 without filtering for writers.
- **IDs:** 113, 236, 431, 590, 652, 351, 432, 854, 905, 917, 1024, 1159, 1165, 1363, 1410, 1425, 1458, 1522, 1569, 1739, 1961, 1966, 1970, 2006, 2059, 2194, 2223, 2340, 2573, 2684, 2780, 3291, 3593, 3635, 3704, 4385, 5000, 5928, 6039, 6195, 6419, 6650, 6811, 6932, 6938, 7087, 7094, 7185, 7264, 7422, 7584, 7863, 8023, 8144, 8318, 8535, 8814, 8821, 9240, 9536, 9562, 9859, 9910, 9984, 10817, 11621, 12099, 12108, 12858.

2. Syntax Errors:

- **Issue:** Submissions contained syntax errors making the queries non-executable.
- **Examples:**
  - Submission 1024: Non-interpretable due to incomplete query `SELECT *;`.
  - Submission 1425: Only contains a semicolon.
  - Submission 1569: Incorrectly counts distinct values.
  - Submission 6932: Incomplete syntax `SELECT FROM;`.
- **IDs:** 1024, 1425, 1569, 1739, 3059, 351, 6932, 8768, 9523.

3. Logical Errors:

- **Issue:** Submissions contained logical errors, such as counting all individuals instead of writers or incorrect usage of SQL clauses.
- **Examples:**
  - Submission 398: Uses wrong column `year` instead of `year_born`.
  - Submission 715: Incorrect use of `GROUP BY` and `HAVING` clauses.
  - Submission 590: Misplaced `GROUP BY` clause.
- **IDs:** 398, 715, 590, 854, 905, 917, 1410, 1427, 1458, 1739, 2780, 3059, 351, 3731, 3853, 4052, 4532, 5242, 5371, 6080, 6419, 6938, 7185, 7422, 7584, 7863, 8023, 8144, 8318, 8535, 8768, 8814, 8821, 9240, 9536, 9562, 9859, 9910, 9984, 10180, 10817, 11031, 11047, 11621, 11973, 12108.

4. Efficiency and Optimization Issues:

- **Issue:** Submissions were logically correct but could be optimized for better performance.

FIGURE 4.2: Snippet of the Generated Report for Exercise 1. (temperature = 0.7)

Observation Type	Occurrences / References	Precision
Incorrect Table Joins or Missing Joins	29/69	0.42
Syntax Errors	5/9	0.56
Logical Errors	20/46	0.43
Efficiency and Optimisation Issues	7/11	0.64
Use of NATURAL JOIN	5/5	1.00
Anomalies/Outliers	3/3	1.00

TABLE 4.16: Results of the manual check and the precision for the report of Exercise 1. (temperature = 0.7)

Due to the amount of manual labelling required we did not verify whether the submissions that were not referenced in the report also fell into these categories highlighted by the LLM. This implies that the recall of the generated reports was not measured.

Impact of Temperature Setting

The report in Figure 4.2 was generated with GPT-4o, with a temperature setting of 0.7. LLMs generate text by predicting the next token based on a probability distribution and the temperature setting influences the model's next token selection process. If it is low, it will more often pick the most likely next token making it more deterministic, and a higher setting makes it choose less likely tokens more often, allowing for more creativity.

To reduce the likelihood of incorrect references to submission ID's, we generated the report again with a lower temperature setting of 0.2. The results for this report in terms of the precision of the references can be found in Table 4.17.

Observation Type	Occurrences / References	Precision
Incorrect Filtering for Writers	8/9	0.89
Syntax Errors	13/13	1.00
Use of Inefficient or Unnecessary Constructs	5/6	0.83
Incorrect Use of JOINs	3/9	0.33
Logical Errors	5/5	1.00
Anomalies or Outliers	4/4	1.00

TABLE 4.17: Results of the manual check and the precision for the report of Exercise 1. (temperature = 0.2).

As we can see, precision was higher with the lower temperature setting of 0.2. However, this setting also retrieved fewer submission IDs overall. This is a classic example of the trade-off between precision and recall: fewer irrelevant IDs were retrieved, but the retrieved ones were highly accurate.

While we did not measure recall exactly due to the manual labelling required, the report generated with a higher temperature setting (Table 4.16) showed many more occurrences. This indicates that while the precision was higher with the lower temperature, the recall likely decreased as fewer relevant submissions were identified.

This analysis shows that adjusting the temperature setting can improve the precision of the identified submission IDs. However, the question still remains whether

providing a few examples of issues is sufficient, or if educators prefer to see a complete list of all submission IDs where these issues are present. The latter would require a different approach to the system.

4.4.5 Process Analysis

The process analysis for Exercise 1 and Student 26 is shown in Figure 4.3. This student made four submissions for this exercise.

The process analysis details each revision (or in this case, each submission) separately and then summaries it at the end. It highlights the student's progress from using 'SELECT *' to 'SELECT id', but that it also still had the issue that it only selects one id, even though the exercise requested to list all possible persons.

Other process analyses presented a similar summary but with slight differences in the structure. For example, some analyses grouped similar revisions instead of describing them individually, which can be preferable for submissions with a large number of revisions.

Due to time constraints, we did not manually verify if the described changes between each revision were accurate. Aggregate reports across students and exercises were also not realised. Still, these analyses were included in the user study in Chapter 5 to gather opinions on the process analysis feature.

4.4.6 Suggested Interventions

Interventions to address the issues were suggested at the bottom of the reports, but seemed to be very generic. A better approach might involve first identifying common issues, and then making a subsequent call to the LLM to obtain interventions based on those observations. This would align more with prompt engineering techniques, which suggests breaking tasks into smaller steps for better results. To evaluate the interventions properly, we refer to the user study in Chapter 5.

4.4.7 Token Usage

An important aspect of using LLMs is the token usage. It is important to address this because there are token limits for API calls. Both input and output tokens count towards this limit. For example, a token limit of 10k would limit the output to 1k tokens if the API call uses 9k tokens in the message input.

We tracked the token usage of the results in this study. Since we loop over all the analyses from the submissions, the token count grows with the number of students. On average, the tokens required for one submission analysis was 630 tokens. In total, there were roughly 315k tokens for the 500 submissions analysed. Since we generated reports for each exercise, the tokens are roughly split over 6 exercises which still fit within the context window of 128k tokens. However, in cases of a higher number of submissions, this may be exceeded.

To address this, we suggest splitting the number of submissions again into smaller chunks and then aggregating the reports generated from these chunks. For example, if there are 1000 submissions, one could divide them into four groups of 250 submissions each. After generating reports for each group, these reports can be combined into one final report. This method could ensure that token usage remains within the limits and the report generation process is still possible.

Initial Review and Analysis of Each Submission

First Submission:

- **Issue:** The query selects all columns (`SELECT *`) rather than just the `id`. The use of nested subqueries and the `EXCEPT` clause adds unnecessary complexity.
- **Improvement Needed:** Simplify the query to directly find the second youngest year of birth and select only the `id`.

Second Submission:

- **Change:** The student changes `SELECT *` to `SELECT id`.
- **Issue:** Although the query now selects only the `id`, it still uses a complex subquery with `EXCEPT`. This approach is not straightforward and could be optimized.
- **Improvement Needed:** Simplify the logic to directly identify the second youngest birth year.

[...]

Progress Evaluation:

- The student demonstrates a clear understanding of the assignment's requirements by progressively simplifying the query and focusing on selecting the `id` only.
- Despite improvements in query simplification, there remains a fundamental issue with using `LIMIT 1`, which needs to be addressed to meet the requirements fully.

Feedback and Recommendations:

1. **First Submission:** Good start, but simplify the logic and avoid `SELECT *`.
2. **Second Submission:** Improvement in selecting only `id`, but the logic still needs simplification.
3. **Third Submission:** Great move towards simplification. However, ensure the query retrieves all relevant rows by handling multiple second youngest persons.
4. **Fourth Submission:** Correct column selection but remove `LIMIT 1` and adjust the logic to retrieve all second youngest persons.

Suggested Correct Query:

```
SELECT p.id
FROM person p
WHERE p.year_born = (SELECT MAX(year_born)
        FROM person
        WHERE year_born < (SELECT MAX(year_born)
                FROM person));
```

FIGURE 4.3: Generated Process Analysis for Student 26 in Exercise 1. We removed the description for the third and fourth submission in this analysis for brevity.

4.4.8 BERT Comparison

We compared the performance of GPT-4o in classifying student submissions to the performance of BERT [29] to assess how well GPT-4o can provide individual analyses that are subsequently used to generate learning analytics.

BERT's superior performance is likely due to its fine-tuning on the dataset, where it was trained on 70% of the data and then classified the remaining 30%. In contrast, we did not fine-tune GPT-4o. As discussed in Chapter 3, our objective was to create a system that is flexible and can work well across a variety of courses without requiring extensive fine-tuning or customisation.

GPT-4o does offer an advantage over BERT in terms of explainability. In the paper in which BERT was used, the authors used Captum [33] to visualise which parts of the submission influence the classification. However, these visualisations, with colour-coded parts (see Figure 2.1), are not always easy to interpret and require technical understanding that might not be accessible to all educators. In contrast, GPT-4o can provide a textual explanation for its decisions, making the rationale behind its classifications easier to understand.

Thus, although fine-tuned models could perform better, GPT-4o can perform decently out-of-the-box while also offering advantages in terms of explainability.

4.5 Discussion

4.5.1 Answer to RQ1

RQ1: How can LLMs be utilised to analyse programming submissions and provide detailed learning analytics in programming education?

We designed a system and formulated two sub-questions to measure the performance:

Sub-question 1 (RQ1a): How accurately can LLMs classify student programming submissions into predefined categories?

Answer: Our findings show that GPT-4o achieved an accuracy of 64% and balanced accuracy of 65% in classifying SQL submissions in four categories. The balanced accuracy indicates that GPT-4o handles the class imbalances better than the baseline, which achieved a balanced accuracy of 37% and accuracy of 66%.

By taking a closer look at the reasoning that the LLM provided for the classifications, we identified areas for improvement that could lead to a higher accuracy. The main observation was that the LLM was very strict, requiring submissions to closely match the model solution even when slight deviations would be acceptable. This resulted in many 'correct' submissions being classified as 'partially correct'. However, this can also be seen as an advantage as this critical look can identify areas for improvement for the students, which could be useful for the generation of analytics in the next step in the system. Another observation was that the LLM missed some context about the specific SQL environment used in the assignments, which it didn't receive in the prompt.

Our primary focus is on the analysis leading to the classification, as this analysis forms the basis for the learning analytics. The classification accuracy serves as an indicator of the reliability of this analysis. Given the accuracy and accompanying reasoning, we believe that GPT-4o can offer a good analysis. Moreover, as LLMs can

only get better, it presents an optimistic prospect for even better performance in the future.

Sub-question 2 (RQ1b): Can LLMs identify common issues and patterns in student programming submissions, and how reliable are these identifications?

Answer: GPT-4o was able to identify several common issues across the submissions and produced reports that highlighted common issues with references to submissions. Initially, only 48% of the referenced submission IDs had the indicated issue due to the limitation of LLMs 'hallucinating' [36]. By adjusting the temperature parameter of the LLM to generate more deterministic reports, we improved the precision of the references to 83%. However, this resulted in lower recall, meaning not all submissions with the identified issue were referenced.

These two sub-questions help us answer RQ1, as this shows that LLMs can be used to generate analytics through a two-step process. First, LLMs analyse individual programming submissions, indicating what is wrong with the submission or providing areas for improvement. In the subsequent step, these individual analyses are consolidated into a single prompt in which the LLM is instructed to generate a report with common issues. The individual analyses can also be used to generate targeted reports per assignment or student. Additionally, the process of a student making an assignment can also be analysed and described by the LLM by including the sequence of submissions in the prompt.

The question still remains if the reports are useful for educators, which we will discuss in the following chapter.

4.5.2 Limitations

Ideally, a labelled dataset of common issues would be used to evaluate the system effectively. However, we did not have access to a dataset that describes all potential issues in an assignment. Manual labelling of such a dataset would also be time-consuming and would require sufficient expertise to ensure accurate evaluation of the submissions, which we lacked.

We considered generating a synthetic dataset using LLMs to introduce predefined errors. However, using the same LLM to both generate and detect the same errors would limit the validity of the evaluation.

A limitation of this study is that the dataset might have been included in the training data of GPT-4o. Since OpenAI does not disclose the datasets used for training, it remains unknown if this is true. If it was used in the training, it could affect the validity of our evaluation since the LLM would have already seen the data.

Another limitation is that the initial prompt describes predefined classes to look for (cheating, correct, non-interpretable, partially correct), which could restrict the model's focus and affects what is eventually presented in the generated reports. For instance, the complexity of the solution might not have been a primary focus but could be relevant to observe. However, it is notable that the LLM did consider complexity in some analyses even though it was not specifically prompted to pay attention to.

Furthermore, the system's performance in other courses or type of assignments is still an open question. Programming assignments in different courses could be more complex and longer, and the effectiveness of the LLM in identifying issues could be different.

Another concern is that the LLM can output texts that are very convincing, even when they may contain incorrect information. The reports and analyses can influence decision-making of the educators, and they should therefore always remain critical when interpreting the information.

Finally, we only manually checked two reports in terms of the precision of the references submission IDs. This limited sample size restricts the generalisability of our findings and more reports should be checked to calculate the precision of the reports. Similarly, the process analyses should have been manually checked for accuracy of describing the process.

Chapter 5

User Study

This chapter presents the user study that was conducted to gain qualitative insights for the system. In Section 5.1, we describe the objective of the user study and in Section 5.2 we describe the methodology, outlining the focus group session. The discussion points and feedback of the participants are presented in Section 5.3. Section 5.4 summarises the common themes that emerged from the feedback. Section 5.5 discusses the implications for the system, outlining specific improvements and acknowledging the study's limitations.

5.1 Objective

Following the system design and evaluation, the objective of the user study was to gather qualitative feedback from educators on the functionality and usability of the system. We aimed to identify areas for improvement for the system and understand the perceived usefulness of the various generated reports. These insights would be valuable to improve the system tailored to their needs.

5.2 Methodology

To achieve this, we designed a focus group with Teaching Assistants (TA), the intended users of the system. This section discusses the focus group details, the outline of the session, and the pilot focus group that was conducted in preparation.

5.2.1 Focus Group

The focus group was designed to gather qualitative feedback from TAs on the various reports contents and the prototype UI that was developed. This includes the course report, exercise reports, submission analyses, suggested interventions, process analyses, and student analyses. The data that was shown in the system was from the same dataset used in Chapter 4 from the Relational Database course at the Australian National University. This dataset contains submissions for exercises in which the students had to write SQL statements.

The study involved a group of 2 former and 3 current TAs at the Faculty of Electrical Engineering, Mathematics and Computer Science at TU Delft. Ideally, TAs from the course that was associated with the dataset would be asked to participate, but this was not feasible since the dataset was from another university. While the TAs that participated were not involved with the course from the dataset, we believed that they could still provide valuable feedback due to their experience as a TA in other programming courses. Furthermore, they all had followed a course in which they learned SQL, so they were familiar with SQL and the types of exercises.

5.2.2 Session Outline

The session was divided into two main parts: the introduction and the discussion. The introduction aimed to familiarise participants with the system, its purpose, and the dataset that was used. The discussion part was designed to gain detailed feedback on various aspects of the system, including the course report, exercise report, submission analysis, suggested interventions, process analysis, and student report. Below is the outline of the session. Appendix B provides a more detailed outline with the questions asked.

Introduction of the session

1. **Welcome:** Introduction to the session.
2. **Overview of the Session:** Brief outline of the session's structure.
3. **Overview of the System:** Explanation of the system's purpose in providing learning analytics to TAs for programming assignments using LLMs, including a description of how it works and the types of reports generated.
4. **Demo of the Prototype:** Demonstration of the system's interface and functionalities, showcasing different reports like student and exercise reports. Appendix A shows the various pages in the user interface. The student report page is also shown in Figure 5.1 for convenience.
5. **Introductory Question:** Participants shared their experience as TAs, focusing on how they identify common student challenges.
6. **Dataset:** Explanation of the course context and dataset, including an example exercise and model solution.

Discussion of the session

Each part of the discussion was intended to last approximately 8 minutes, focusing on specific aspects of the system. For each part, the prototype of the system was shown with the generated reports based on the aforementioned dataset, and a question was asked about the strengths and weaknesses of the report. The structure is outlined below:

1. **Course Report (8 minutes):** Presentation of common issues and challenges from all course submissions.
2. **Exercise Report (8 minutes):** Presentation of common issues and challenges from submissions in specific exercises.
3. **Submission Analysis (8 minutes):** Analysis of individual submissions.
4. **Suggested Interventions (8 minutes):** Presentation of suggested interventions for the course and exercises.
5. **Process Analysis (8 minutes):** Analysis of the process with varying numbers of submissions.
6. **Student Report (8 minutes):** Reports generated based on all submissions made by a student throughout the course.

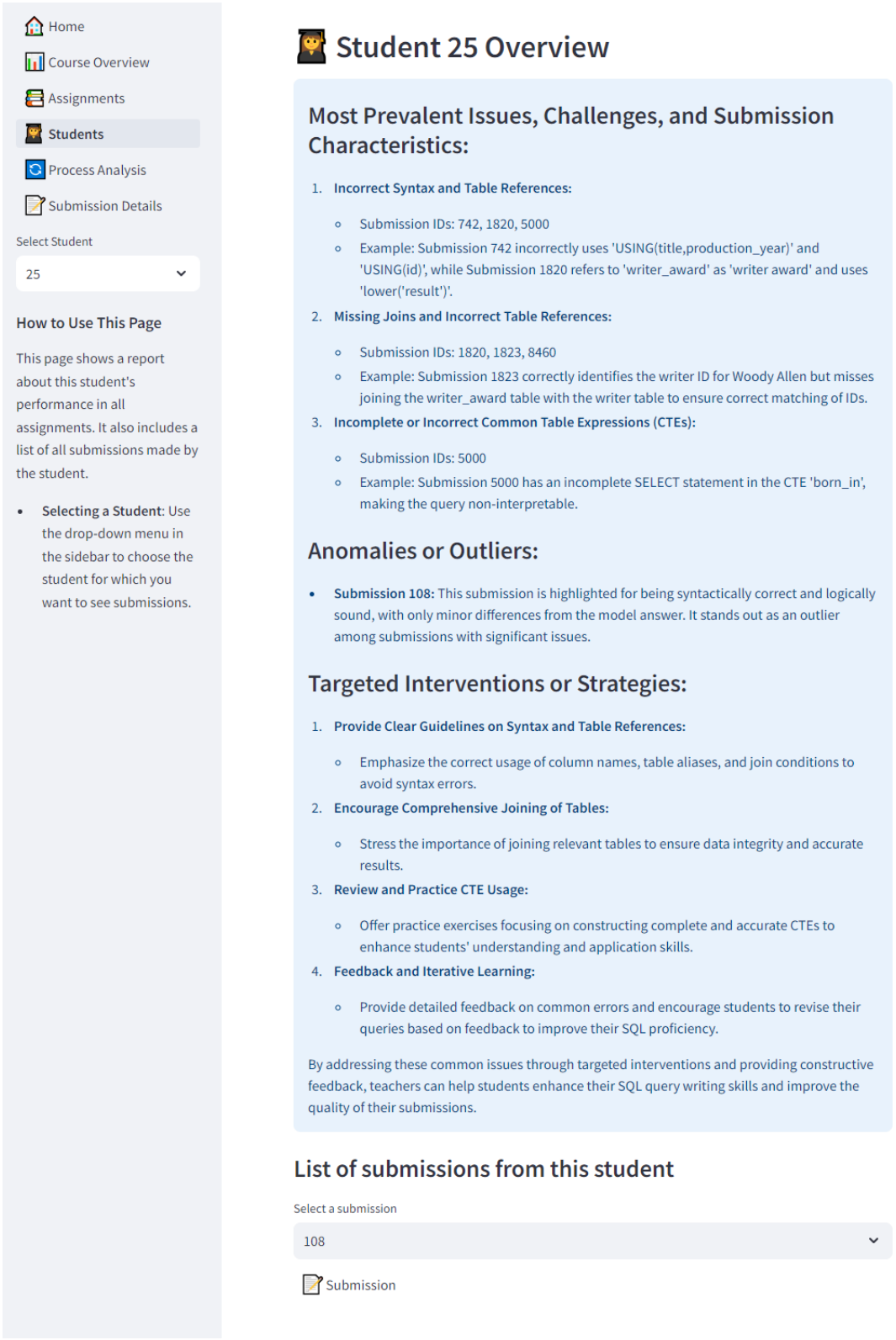

FIGURE 5.1: User Interface Student Overview that was shown in the focus group.

5.2.3 Pilot Study

A pilot study was conducted to identify improvements that could be made in the structure of the session. This study included two student participants and lasted an hour.

The pilot study revealed that discussions could naturally overlap between different parts of the session outlined in the previous subsection. It was found to be more effective to allow the conversation to flow naturally rather than strictly adhering to the outline. For example, naturally transitioning from a report to specific submissions that were listed was allowed since these go hand-in-hand.

Furthermore, although the participants were not TAs, they provided valuable insights into the tool. Some discussion points and feedback from the pilot study will be mentioned in the analysis section, as there was a lot of agreement on various points between the pilot study participants and the actual focus group.

5.3 Analysis of Findings

This section describes the discussions and comments made by the participants. The focus group session lasted an hour which fell into the outlined time frame we intended.

5.3.1 Experience as a TA

This part of the discussion provided insights into the experiences of the participants as a TA for the courses they assisted. They revealed some preferences in the way they work as a TA and shared different perspectives on their roles, highlighting their challenges and also methods they find most effective or preferable.

One participant mentioned that they have a preference for courses that allow for the automation of grading or feedback processes. They explained their use of scripts to scan for specific keywords or to help them fill in a checklist when reviewing students' work.

Another participant expressed their preference for project-based courses. They indicated that they rely on statistics, such as lines of code written per students, to get an idea of how much students contribute to group projects. They also indicate that in another course they cared less about statistics and more about discussions with the students to assess if they know what they are talking about. They also mentioned that they do not focus on analysing submissions for courses that don't have mandatory assignments, since not many students would make those assignments.

5.3.2 Course Report

In this part of the discussion, participants evaluated the 'Course Report', which provides a summary of common challenges and issues identified across all submissions in the course. Overall, the participants liked that the system could identify common issues, but they proposed several improvements to make the report more effective. They wanted more details on the frequency of issues, hyperlinks in the reports, clearer ranking of issues, expandable sections, and customisability.

Detail on Issue Frequency

Their first comment they made was the desire for more detailed information on the frequency of specific issues. It was unclear in how many submissions out of all of the submissions an issue was identified. They would have preferred exact numbers or percentages alongside the mentioned issue to gauge how widespread each issue is. This suggestion was also mentioned by the participants in the pilot study.

Hyperlinks to Submissions

Another suggestion that was stressed in both the pilot and focus group was the inclusion of hyperlinks to specific submissions where issues were identified. In the prototype this feature was not realised, but would make it a lot simpler to navigate to a specific submission.

Ranking of Issues

The ranking of the issues was also discussed and they felt that it was unclear what the ranking was based on. They mentioned that a ranking based on severity would help prioritise which issues need immediate attention and which are less critical. A ranking based on frequency would for example be less significant if the most common issues are small typos.

Expandable Sections

Furthermore, the participants found that the report was quite overwhelming due to the amount of text. To make the report less overwhelming, participants recommended using collapsible and expandable sections in the report. The common issues would then be written in a few words, which the user could expand to find details, examples, and submissions where this issue could be found. When asked about the submission IDs, they mentioned that they prefer to have a collapsible list of all relevant submission IDs, instead of only a few examples. The participants also mentioned that some sort of visuals or graphs would help them get a good overview more quickly, reducing the feeling of being overwhelmed by the text.

Customisability

Lastly, the participants suggested that the system should allow TAs or teachers to customise the report based on their specific needs for the course. They could then choose to focus on certain types of issues.

5.3.3 Exercise Report

Reviewing the exercise reports, the participants mentioned the same improvements as for the course report, since the reports were similar in structure.

One additional remark was the need for a consistent format of the reports as the reports were slightly different for each exercise due to the stochastic nature of LLMs. They did not mention a preference for a specific format, as long as it is consistent. A more standardised format would make it easier to compare and analyse the reports across different exercises. The participants of the pilot study also indicated the need for a consistent format.

5.3.4 Submission Analysis

During the discussion on the submission analysis, the participants gave feedback to increase both the readability and the effectiveness. The trust they have in the system was also briefly discussed.

Structure of Text

Several participants noted that the analysis text provided a lot of text which could be overwhelming, similar to the previous reports. They suggested a more concise analysis by presenting the text in bullet points for better readability. A participant mentioned that a checklist format would be nice so TAs could quickly see if key criteria for that exercise were met. They suggested that the LLM could generate a consistent template or rubric, which would then be used to analyse all submissions.

Additionally, they would like to quickly see an indication of how good or bad the submission was with a numeric score for example. They draw comparisons with WebLab, a learning management system designed for programming assignments [68]. In WebLab, TAs can quickly see how well the submission was made by means of a score based on automated tests.

Suggesting Improvements and Fixes

Participants would also like the idea of the submission analysis not only identifying errors, but also suggesting improvements and fixes. While the exercise and course reports do provide suggestions to address common issues, suggestions on an individual submission basis would also be beneficial for them if they would have to assist the student that is stuck on the exercise. They emphasised that the suggestions would need to be specific and provide explanations for why certain fixes are recommended.

Trust in Generated Analysis

There was a concern from one participant about the accuracy of the analysis generated by the LLM. They expressed that they were unsure how much they would trust what is written by the LLM, and would like to have an easy way to verify it. Another participant asked which LLM was used for the analysis, and after the facilitator specified that GPT-4o was used, they said that they would trust the assessment made by that model.

5.3.5 Suggested Interventions

Participants generally appreciated the inclusion of suggested interventions in the reports, but they pointed out that some suggestions were too obvious or vague, requiring the need for more specific suggestions.

Specific Feedback and Recommendations

Participants noted that the interventions provided should be more specific to the course or assignment. For example, instead of a general suggestion like "clarify assignment requirements" it would be more helpful to indicate exactly which part of the assignment requirements were unclear and suggest ways to improve them.

Another example given that needed to be more specific was one where the system recommended to give training on SQL syntax, which was found to be too obvious. Here the system could suggest specific topics that need to be addressed.

Integration with Course Evaluations

A participant brought up the idea of integrating the suggested interventions with the outcomes of course evaluations that are conducted at the end of a course. For example, if the course evaluation indicates that students found that lectures were not engaging, the system could take this feedback into consideration and suggest more interactive in-class activities.

Ranking of Suggestions

Similar to the ranking of issues, participants suggested that the interventions could be ranked by their importance or potential impact. This would help teachers and TAs prioritise the interventions.

5.3.6 Process Analysis

The process analysis shown to the participants presented the process of a student making a single exercise. The participants discussed various ways to improve it by for example considering the process throughout the course instead of the process in a single exercise. This subsection highlights their perspectives.

Relevance to TA Workflows

There was a short discussion about the relevance of the process analysis at different educational levels. One participant thinks that such detailed feedback might be more useful in a high school setting where students need more guidance. In a master's level course, they find that they generally do not need to dive into each process in such detail unless specifically requested by the student. Instead, they focus more on the final submission and overall performance. Additionally, one participant suggested that it could be more useful to show the analysis to the student instead of a TA, such that the student can reflect on it themselves.

Analysing Processes Differently

One participant suggested a slight enhancement that could be made to better highlight whether students have learned from their mistakes over time. The system could analyse whether specific issues identified in earlier exercises are corrected in later ones, thus providing insights into the student's learning process. Opposed to focusing on the process of multiple submissions for a single exercise, this could provide a more useful analysis.

Format

Similar to the comments made for the other reports, the participants would like to see improvements in the way the analysis is presented in order to improve readability. They suggested better summarising and highlighting key changes, with an option to expand for more details. A consistent format for each analysis would also be more beneficial here. This would make it easier for TAs to quickly assess the

process without being overwhelmed with too much detail. Graphs or other visual representations was also mentioned as an improvement that could help get a better overview and easier understanding.

5.3.7 Student Report

The student reports were similar to other reports and the participants indicated similar improvements to those mentioned in the other reports. These feedback comments include a consistent format and expandable sections for a better overview.

5.4 Common Themes

Throughout the focus group, several common themes emerged regarding the participants' feedback and suggestions for improving the system. Overall, they liked the idea of the system and thought it is a good base, but found that it needed some improvements to make it more usable for TAs.

5.4.1 Structure of Reports

The need for a standardised format across all reports was a common theme. Participants indicated that a consistent structure would make it easier to compare and analyse information across different exercises and submissions.

Furthermore, they would prefer a more concise presentation of the contents of the reports where the details are collapsed and able to be expanded when the user would want to. Bullet points especially would make it more pleasant to read. This would ensure they are not overwhelmed with a lot of text.

They also indicated the desire for exact numbers or percentages of the prevalence of issues to get a better impression of the magnitude of observations. Additional metrics or graphs would also enhance the reports, and when referencing submissions, a hyperlink to the submission would make it a lot more user-friendly.

5.4.2 Ranking

The ranking of both the issues identified and the suggested interventions were unclear to the participants. They suggested either making it clear that it is based on the frequency of the issue identified, or rank it based on the impact or severity of the issue and suggestion. Another option was to simply use bullet points instead of numbered lists to indicate there is not a specific ranking, since the numbers implied they were ranked.

5.5 Discussion

The participants provided a multitude of suggestions to improve the system, focusing mainly on how it is presented to the user. The pilot group had similar suggestions as the focus group regarding the readability of the report.

5.5.1 Implications for the System

We discuss the implications of the feedback from the focus group for the system:

- **Consistent Theme and Expandable Sections:**

  Determining the level of detail to display in learning analytics dashboards for different stakeholders is an open issue described by Schwendimann et al. [18] in their literature review, as users could get overwhelmed by the amount of information presented to them. The participants in our study highlighted the need for a consistent theme, bullet points, and expandable sections in the report. This can be achieved by establishing a predefined output structure and including it in the prompt. However, collapsible and expandable sections are not included in the syntax for Markdown, which is the format the LLM typically responds with. Expandable sections could be realised with HTML tags `<details>` and `<summary>`, which some Markdown parsers do allow. Streamlit, the framework used for the prototype, does support this.

- **Hyperlinks:**

  Adding hyperlinks was not supported by Streamlit, and hence, this feature was not included. In an LMS like WebLab that does support links, hyperlinks could be added after generation of the reports by a script that replaces submission IDs with submission IDs with links.

- **Additional Context:**

  Participants suggested the inclusion of additional context, such as rubrics. In the System Design chapter this was discussed as something that can be included, but the dataset used did not contain a rubric. In courses where rubrics are available, the analysis prompt can be modified to include this additional context.

- **Customisability:**

  Participants suggested allowing users to adjust prompts before running analyses, enabling them to specify what the LLM should focus on when analysing submissions. To implement this, a user interface component can be added, providing the option to modify the default prompt before running the analyses. To maintain the system's design consideration of minimising the need for extensive modifications for each course, this feature should remain optional.

- **Trust in the System:**

  One participant expressed their concern on the trustworthiness of the system's analysis. In the system design, we explained how we aim to address this by instructing the LLM to provide reasoning for each part of the analysis and report. The comment made by the participant stressed that the observations made by the LLM should be well-reasoned such that the user could verify it.

- **Process Analysis:**

  Participants noted that they do not typically examine the process behind submissions. While this might indicate that there is no need for such a feature, it might also indicate that the systems they used as a TA did not facilitate such analysis properly. This feature, along with the proposed improvements, could encourage TAs to consider this aspect, ultimately leading to a better learning experience for students.

- **Visualisations:**

Some participants expressed a desire for more visualisations to support the text. This was also remarked in existing research, where teachers were presented with dashboards and some would like visuals accompanying the written text report [69]. The participants in the focus group did not specify the exact shape or form visualisations should take. To define these requirements, further focus groups or interviews with TAs would be necessary.

5.5.2 Answer to RQ2

RQ2: How do educators perceive the usefulness of LLM-generated learning analytics?

Answer: Teaching assistants that participated in the focus group provided many suggestions to improve the system, primarily focusing on the presentation of the learning analytics. The emphasis was on improving the readability of the reports. They expressed the need for a consistent report structure, bullet points, and expandable sections to make the reports less overwhelming. Additionally, some participants wanted visualisations to complement the reports, making them easier to interpret quickly.

Participants also noted that the suggested interventions were too generic. They recommended making the suggestions more specific, for example by including information about the exercise or course setup in the prompt. Overall, the focus group found that the system could be useful, but indicated various aspects that need to be improved to enhance the usability of the reports.

5.5.3 Limitations

The focus group did not comment on the quality of the findings presented in the reports in terms of the relevance of the observations. This is a limitation, but is understandable since the participants were not involved in the course related to the dataset and were only presented with the course content right before discussion into the system. Thus, they could not provide an informed opinion on the quality. Ideally, the system would be evaluated with TAs from the corresponding dataset or course.

Additionally, the focus group did not include head teachers or professors, who might have provided different insights and perspectives. Future research should include interviews with these stakeholders to understand their needs. In retrospect, such interviews and focus group sessions should also have been conducted before the system design. However, it might have been challenging to convey the system's potential and possible insights without first designing a prototype using actual data.

The size of the focus group was also relatively small, which limits the generalisability of the findings. The feedback obtained was valuable, but conducting additional focus group sessions with other participants could provide a better understanding of the system's strengths and weaknesses. More participants would also help validate the insights, if they agree on the same suggestions.

Furthermore, the participants did not use the system for an extended period. A study in which participants would use the system for a long session by themselves or throughout a course could lead to better qualitative feedback. This would provide a better understanding of the system's usefulness in a real educational setting.

Lastly, the use of only one dataset also presents a limitation. The dataset was specific to a relational database course with relatively short submissions. Courses with longer and more complex programming submissions might present different

challenges and subsequently different feedback for the design of the system. Future research should include datasets from different courses to identify this.

Chapter 6

Discussion

In this chapter, we synthesise the findings from the previous chapters, discuss the overall implications, reflect on the main limitations, and propose recommendations for future work.

6.1 Answer to Research Questions and Implications

We aimed to explore the use of LLMs in analysing programming submissions and providing learning analytics. We developed a system and focused on two primary research questions, where RQ1 had two sub-questions:

- **RQ1: How can LLMs be utilised to analyse programming submissions and provide detailed learning analytics in programming education?**
  - **RQ1a: How accurately can LLMs classify student programming submissions into predefined categories?**

    The results indicated that GPT-4o achieved an accuracy of 64% and balanced accuracy of 65% in classifying submissions in an SQL dataset, surpassing the baseline balanced accuracy of 37%. The accompanying analysis behind these classifications revealed that GPT-4o could be better instructed to achieve higher accuracy, but also that the GPT-4o was very critical, identifying areas for improvement in correct submissions. As the individual analyses form the basis for generating analytics, the classification accuracy serves as an indicator of the reliability of the analyses. Given the accuracy and accompanying reasoning, we believe that GPT-4o can offer a good analysis that can be used in aggregated reports. Moreover, as LLMs continue to improve, it presents an optimistic prospect for even better performance in the future.
  - **RQ1b: Can LLMs identify common issues and patterns in student programming submissions, and how reliable are these identifications?**

    GPT-4o was able to identify common issues in SQL assignment submissions, which are summarised in reports that contain references to submissions. Initially, the references to specific submissions were imprecise due to the limitation of LLMs 'hallucinating' [36]. By adjusting the temperature parameter of the LLM, precision improved significantly from 48% to 83%. However, this resulted in lower recall, indicating that not all submissions with the identified issues were referenced.

  These sub-questions help answer RQ1, as this shows that LLMs can be used to generate learning analytics through a two-step process. First, the LLM analyses individual programming submissions, indicating what is wrong with the

submission or providing areas for improvement. In the subsequent step, these individual analyses can then be consolidated into a single prompt in which the LLM is instructed to generate a report with common issues. The individual analyses can also be used to generate targeted reports per assignment or student. Additionally, the process of a student making an assignment can also be analysed and described by the LLM by including the sequence of submissions in the prompt.

- **RQ2: How do educators perceive the usefulness of LLM-generated learning analytics?**

  In the focus group, teaching assistants found the system potentially useful but emphasised the need for better report readability, structure, and visualisations. They suggested that reports should include percentages of the prevalence of issues, along with a complete list of submissions where each issue is present. Additionally, some participants felt that the suggested interventions were too generic and could be related more to the exercises or overall course setup.

Based on our system evaluation and user study, we identified a key implication for the system. While the system was able to identify common issues and reference submissions with high precision, the evaluation also revealed that many relevant submissions were not referenced, indicating low recall. Additionally, the user study highlighted the need for concrete percentages and numbers, along with a complete list of submissions where specific issues are present. Our current system design lacked this aspect, and would need to be addressed to enhance the learning analytics.

Furthermore, the user study highlighted the need to improve the structure and readability of the analyses and reports. Participants suggested a consistent theme, bullet points, and expandable sections. This can be achieved by predefining the output structure in the LLM prompt, ensuring that reports are easier to read and less overwhelming.

Additionally, the suggested interventions were perceived by some participants as too generic. To make the suggestions more specific, they should be related more to the assignment details or to the course setup. This would increase the actionability of the learning analytics.

6.2 Limitations

Many limitations in each of the three previous chapters were already described. In this section, we discuss the main limitations.

One of the primary limitations is the challenge associated with finding a suitable dataset for evaluation. The ideal evaluation requires a dataset of programming submissions that are labelled with the specific issues of the submission. However, access to these datasets are limited, and manually labelling the data is time-consuming and also requires relevant expertise.

Another significant limitation is that the user study was not conducted with educators who were familiar with the course assignments. The system would be best evaluated by the educators related to the course from which the data in the system is derived. In the focus group, the participants provided valuable feedback but could not comment in-depth on the significance or relevance of the identified issues. Additionally, the participants did not use the system over an extended period. A study

where participants use the system for a long session by themselves or throughout a course could lead to better qualitative feedback. A more targeted user study with both a quantitative and qualitative aspect would also be appropriate to measure the usefulness of the system.

Lastly, the study was conducted only with a dataset of SQL assignments and submissions. These SQL submissions are relatively short compared to other programming assignments in different courses with different programming languages. The effectiveness of the system in identifying issues in larger submissions and across various programming languages remains uncertain.

6.3 Future Work

Based on the discussion and limitations, we make the following recommendations for practical implementation and further research:

To address the concerns about accurate statistics and references to submission, we propose a solution where a second pass over the submissions is done. After identifying 5 (or more) common issues in the submissions as we currently do in the system, the LLM could be instructed to create unique tags for these issues. In the second pass, the LLM would then be instructed to indicate whether each of the 5 issues is present in each submission. For example, the LLM would evaluate each submission and mark each issue as true or false:

Submission	Issue 1	Issue 2	Issue 3	Issue 4	Issue 5
Submission 1	True	False	True	False	True
Submission 2	False	True	False	True	False
Submission 3	True	True	False	False	True
Submission 4	False	False	True	True	False

TABLE 6.1: Example of Second Pass Analysis for Common Issues.

We can then programmatically count occurrences, calculate statistics, and keep track of which submissions had which issues. In the user interface, these can then be, for example, shown next to the reports. As discussed in Chapter 3, predefining these tags ourselves could limit what the LLM would look for, potentially overlooking issues that were not anticipated. In this proposed solution, that problem is mitigated by allowing the LLM to dynamically generate tags based on the issues it identifies in the submissions.

While this could improve the system, future studies could also directly focus on evaluating the system with educators who are directly related to the courses from which the data is derived. These educators can provide more in-depth feedback on the significance and relevance of the identified issues in the reports. Conducting studies where participants use the system for a longer period would also provide valuable insights into its usefulness.

Furthermore, to ensure the system's generalisability it is important to conduct studies with different programming assignments and languages. This includes evaluating the system with larger and more complex programming tasks and submissions. If research shows that the system performs effectively across various programming languages and more extensive assignments, it can be a helpful addition to existing learning management systems that handle these different assignments in one system.

Chapter 7

Conclusion

This thesis aimed to addressed the gap in the field of programming education concerning the potential use of LLMs in generating learning analytics. While many learning analytics systems exist, they often lack the capability to provide actionable insights. Therefore, we designed a system leveraging LLMs, where submissions are first individually analysed, and the analyses are subsequently aggregated to identify common issues. Through a system evaluation and user study, this thesis demonstrated the potential of the system in supporting educators with learning analytics. Specifically, our findings indicate that LLMs can analyse submissions for SQL assignments with a reasonable degree of accuracy and can identify common issues. Educators found these insights potentially useful but identified aspects in the reports that should be refined.

The study is limited by its reliance on a dataset from a single course, the lack of input from stakeholders familiar with that course, and the absence of a study in which educators use the system over a longer period. Despite these limitations, this thesis provides a solid foundation and useful insights for future work in using LLMs for learning analytics.

To enhance the system further, we recommend investigating methods to generate accurate statistics and complete references to submissions in reports, addressing the requests of participants in the user study. Future research should involve educators who are familiar with the course content to provide more targeted feedback and evaluate the system's usefulness over a longer period. Additionally, investigating the system's effectiveness with different programming languages and more complex assignments will help determine the system's generalisability.

Appendix A

User Interface

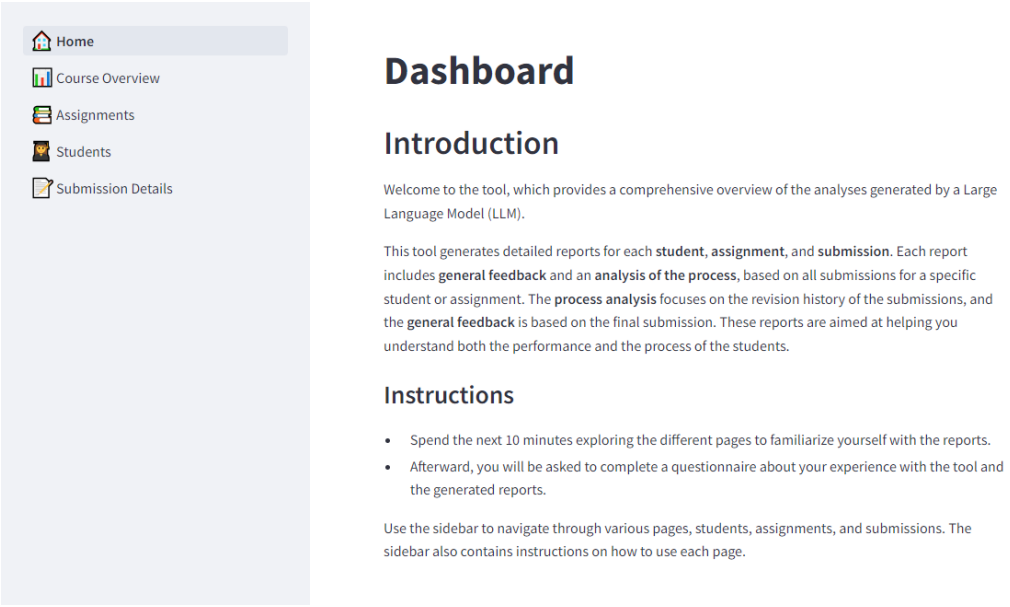

FIGURE A.1: User Interface Main Overview

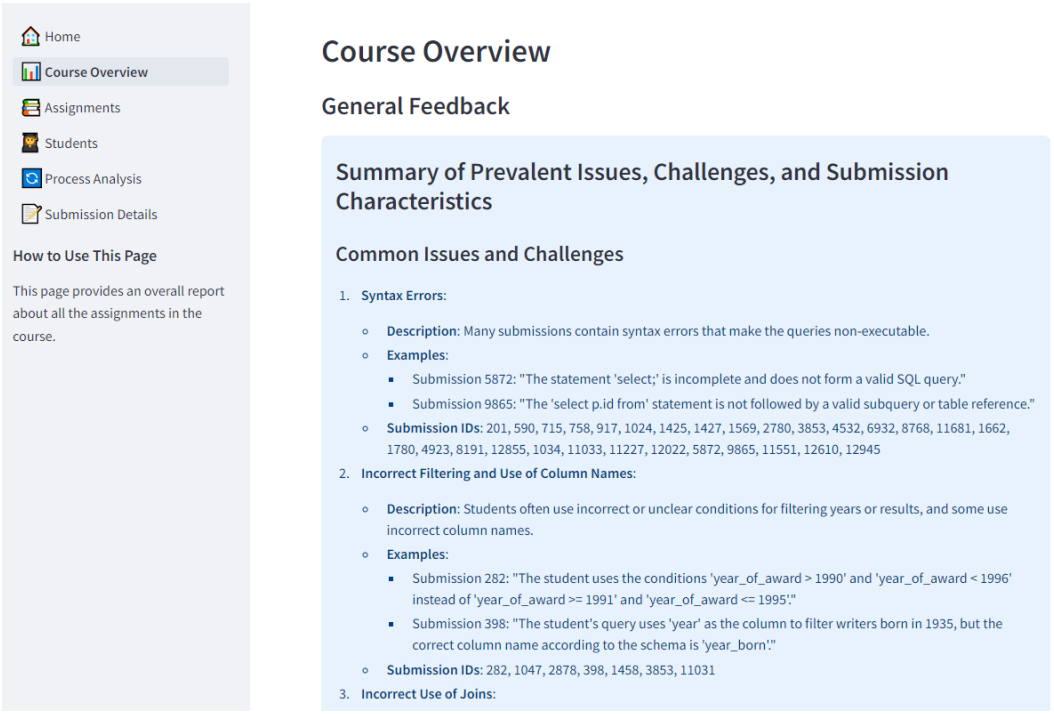

FIGURE A.2: User Interface Course Overview

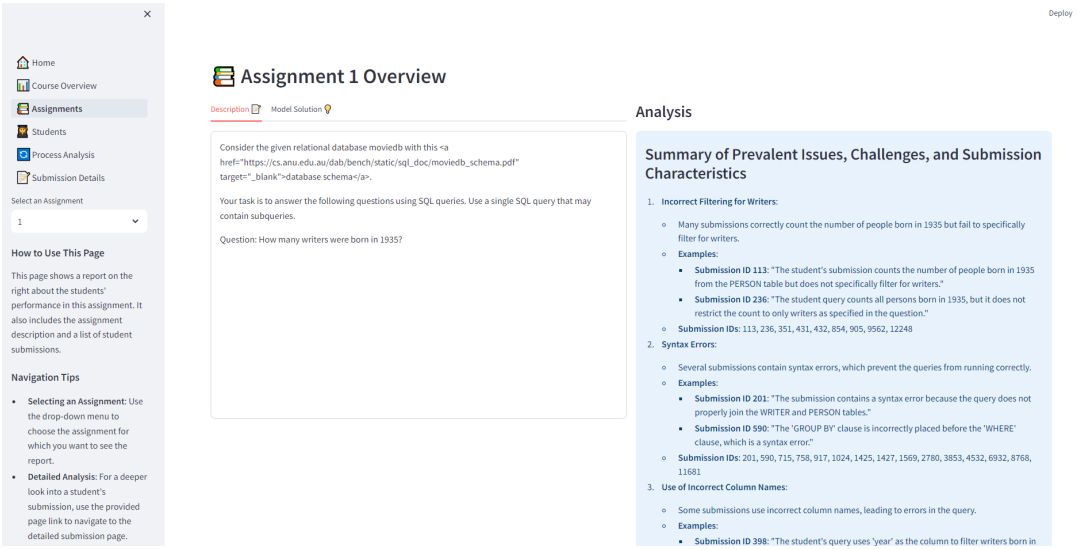

FIGURE A.3: User Interface Assignment Overview

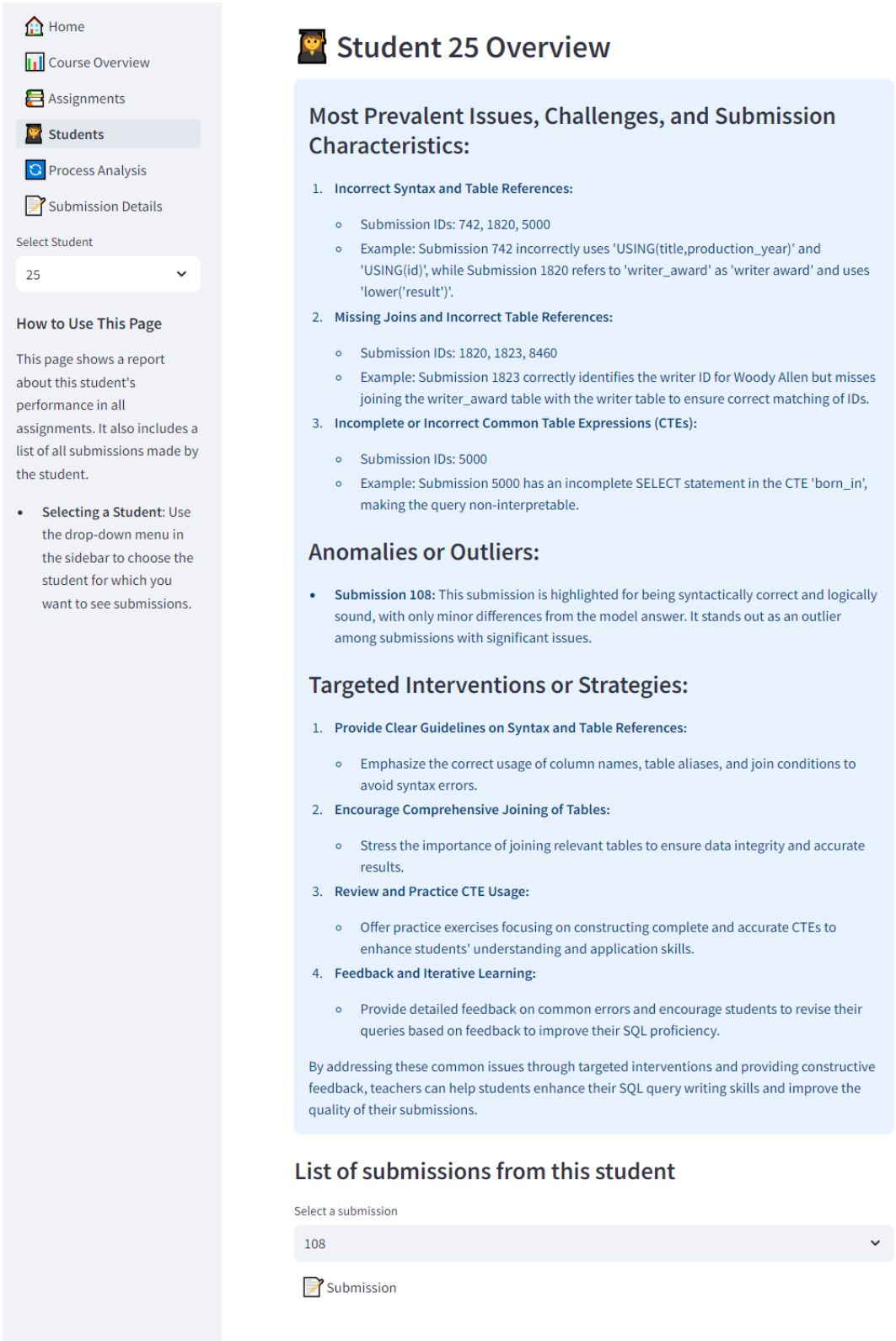

FIGURE A.4: User Interface Student Overview

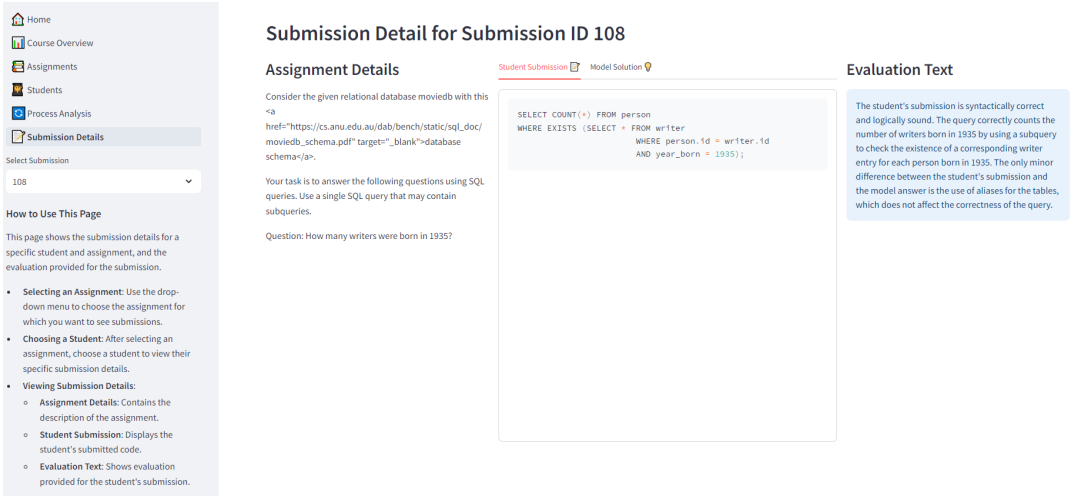

FIGURE A.5: User Interface Submission Details

Appendix B

Focus Group Session Outline

B.1 Introduction (10 minutes)

1. **Welcome:** The session began with a an introduction of what this session was about.

2. **Overview of the Session:** A brief overview of the session's structure was provided to set expectations.

3. **Overview of the System:** An introduction to the system was presented, explaining its purpose in providing learning analytics to TAs for programming assignments using LLMs. A short description of how the system works was also given to explain how it makes use of LLMs. The types of reports were also mentioned.

4. **Demo of the Prototype:** A quick demonstration of the prototype was conducted to familiarise participants with the system's interface and functionalities. This demo showcased the user interface and the types of reports generated, showing the different reports like student and exercise reports.

5. **Introductory Question:** Participants were asked about their experience as TAs, specifically focusing on if and how they identify common struggles and challenges among students in the courses they assisted. This question aimed to understand their background.

6. **Dataset:** Before starting with the discussion, the context of the course and the dataset used for the system was explained. An example exercise was shown along with the model solution to familiarise participants with the type of exercises and content they would see in the system.

B.2 Discussion (48 minutes)

Each part of the discussion was intended to last approximately 8 minutes, focusing on specific aspects of the system. For each part, the prototype of the system was shown with the generated reports based on the aforementioned dataset, and a question was asked about the strengths and weaknesses of the report. The structure is outline below, also indicating the reason why these questions were asked.

1. **Course Report (8 minutes):**

   The course report was shown that present the common issues and challenges from all submissions made in the course.

Question: What are the strengths and weaknesses of the course report? Are you missing anything?

Why: To understand the participants' views on the overall usefulness of the course report, and to identify any improvements that could be made, for example in the way it is presented.

2. **Exercise Report (8 minutes):**

   Multiple exercise reports were shown that present the common issues and challenges from all the submissions in the respective exercise.

   Question: What are the strengths and weaknesses of the exercise report? Would you prefer seeing all related submission IDs or just examples?

   Why: This was asked to again understand the participants' views on the overall usefulness of the reports, and to identify any improvements that could be made. The question about the submission IDs arose from the system evaluation, in which we described the problem of submission IDs being referenced that were not relevant.

3. **Submission Analysis (8 minutes):**

   Multiple submissions were shown with the analysis generated for those submissions.

   Question: What are the strengths and weaknesses of the individual submission analysis?

   Why: To understand the participant's views on the analysis made by the LLM on the individual submissions.

4. **Suggested Interventions (8 minutes):**

   The suggested interventions were shown for the full course and a few exercises.

   Question: What are the strengths and weaknesses of the suggested interventions?

   Why: This was asked to gather the participants' opinions on the suggested interventions to determine their practicality and usefulness in addressing common issues identified in student submissions.

5. **Process Analysis (8 minutes):**

   Multiple analyses were shown, each with a varying number of submissions that were made. E.g. some analyses included the process based on only 2 submissions that the student needed to make, and some included more than 10 submissions throughout the making of 1 exercise.

   Question: What are the strengths and weaknesses of the process analysis? Do you think it would be useful?

   Why: This was evaluated to understand what they think of the report and understand its potential in tracking and analysing the process of a student making an exercise, identifying where they struggled.

6. **Student Report (8 minutes):**

   Multiple reports were shown with the report generated based on all the submissions a student made throughout the course.

Question: What are the strengths and weaknesses of the student reports? Do you think it would be useful?

Why: This question aimed to gather feedback on the student report's perceived usefulness and effectiveness in providing an overview of a student's performance across multiple exercises.

Bibliography

[1] E. National Academies of Sciences, *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. Oct. 2017, ISBN: 978-0-309-46702-5. DOI: 10.17226/24926.

[2] *The Challenges of the Enrollment Surge for the Unit*, [Online; accessed 14. Jun. 2024], Feb. 2017. [Online]. Available: https://cra.org/data/generation-cs/impact-on-the-unit.

[3] *Penetrating the Fog: Analytics in Learning and Education*, [Online; accessed 3. Jun. 2024], Sep. 2011. [Online]. Available: https://er.educause.edu/articles/2011/9/penetrating-the-fog-analytics-in-learning-and-education.

[4] J. C. Paiva, J. P. Leal, and A. Figueira, "Automated assessment in computer science education: A state-of-the-art review," *ACM Trans. Comput. Educ.*, vol. 22, no. 3, 2022. DOI: 10.1145/3513140. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/3513140.

[5] K. M. Ala-Mutka, "A survey of automated assessment approaches for programming assignments," *Computer Science Education*, vol. 15, no. 2, pp. 83–102, 2005. DOI: 10.1080/08993400500150747. eprint: https://doi.org/10.1080/08993400500150747. [Online]. Available: https://doi.org/10.1080/08993400500150747.

[6] C. Klein, J. Lester, H. Rangwala, and A. Johri, "Technological barriers and incentives to learning analytics adoption in higher education: insights from users," *J. Comput. High. Educ.*, vol. 31, no. 3, pp. 604–625, Dec. 2019, ISSN: 1867-1233. DOI: 10.1007/s12528-019-09210-5.

[7] B. Wong and K. Li, "A review of learning analytics intervention in higher education (2011–2018)," *Journal of Computers in Education*, vol. 7, May 2019. DOI: 10.1007/s40692-019-00143-7.

[8] B. Rienties, S. Cross, and Z. Zdrahal, "Implementing a learning analytics intervention and evaluation framework: What works?" In *Big Data and Learning Analytics in Higher Education: Current Theory and Practice*, B. Kei Daniel, Ed. Cham: Springer International Publishing, 2017, pp. 147–166, ISBN: 978-3-319-06520-5. DOI: 10.1007/978-3-319-06520-5_10. [Online]. Available: https://doi.org/10.1007/978-3-319-06520-5_10.

[9] *ChatGPT*, [Online; accessed 14. Jun. 2024], Jun. 2024. [Online]. Available: https://openai.com/chatgpt.

[10] Z. Xiao, X. Yuan, Q. V. Liao, R. Abdelghani, and P.-Y. Oudeyer, "Supporting qualitative analysis with large language models: Combining codebook with gpt-3 for deductive coding," in *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces*, ser. IUI '23 Companion, Sydney, NSW, Australia: Association for Computing Machinery, 2023, 75–78, ISBN: 9798400701078. DOI: 10.1145/3581754.3584136. [Online]. Available: https://doi.org/10.1145/3581754.3584136.

[11] J. Finnie-Ansley, P. Denny, B. Becker, A. Luxton-Reilly, and J. Prather, "The robots are coming: Exploring the implications of openai codex on introductory programming," Feb. 2022, pp. 10–19. DOI: 10.1145/3511861.3511863.

[12] I. Azaiz, O. Deckarm, and S. Strickroth, "Ai-enhanced auto-correction of programming exercises: How effective is gpt-3.5?" *International Journal of Engineering Pedagogy (iJEP)*, vol. 13, no. 8, 67–83, Dec. 2023, ISSN: 2192-4880. DOI: 10.3991/ijep.v13i8.45621. [Online]. Available: http://dx.doi.org/10.3991/ijep.v13i8.45621.

[13] D. Clow, "The learning analytics cycle: Closing the loop effectively," in *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, ser. LAK '12, Vancouver, British Columbia, Canada: Association for Computing Machinery, 2012, 134–138, ISBN: 9781450311113. DOI: 10.1145/2330601.2330636. [Online]. Available: https://doi.org/10.1145/2330601.2330636.

[14] X. Du, J. Yang, B. E. Shelton, J.-L. Hung, and M. Zhang, "A systematic meta-review and analysis of learning analytics research," *Behaviour & Information Technology*, vol. 40, no. 1, pp. 49–62, 2021. DOI: 10.1080/0144929X.2019.1669712. eprint: https://doi.org/10.1080/0144929X.2019.1669712. [Online]. Available: https://doi.org/10.1080/0144929X.2019.1669712.

[15] O. Zawacki-Richter, V. I. Marín, M. Bond, and F. Gouverneur, "Systematic review of research on artificial intelligence applications in higher education – where are the educators?" *Int. J. Educ. Technol. High. Educ.*, vol. 16, no. 1, pp. 1–27, Dec. 2019, ISSN: 2365-9440. DOI: 10.1186/s41239-019-0171-0.

[16] M. Hussain, W. Zhu, W. Zhang, and S. M. R. Abidi, "Student Engagement Predictions in an e-Learning System and Their Impact on Student Course Assessment Scores," *Comput. Intell. Neurosci.*, vol. 2018, Oct. 2018, ISSN: 1687-5265. DOI: 10.1155/2018/6347186.

[17] B. T.-m. Wong and K. C. Li, "A review of learning analytics intervention in higher education (2011–2018)," *J. Comput. Educ.*, vol. 7, no. 1, pp. 7–28, Mar. 2020, ISSN: 2197-9995. DOI: 10.1007/s40692-019-00143-7.

[18] B. A. Schwendimann, M. J. Rodríguez-Triana, A. Vozniuk, *et al.*, "Perceiving learning at a glance: A systematic literature review of learning dashboard research," *IEEE Transactions on Learning Technologies*, vol. 10, no. 1, pp. 30–41, 2017. DOI: 10.1109/TLT.2016.2599522.

[19] M. Valle Torre, C. Oertel, and M. Specht, "The sequence matters in learning - a systematic literature review," in *Proceedings of the 14th Learning Analytics and Knowledge Conference*, ser. LAK '24, ACM, Mar. 2024. DOI: 10.1145/3636555.3636880. [Online]. Available: http://dx.doi.org/10.1145/3636555.3636880.

[20] P. Blikstein, M. Worsley, C. Piech, M. Sahami, S. Cooper, and D. Koller, "Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming," *Journal of the Learning Sciences*, vol. 23, no. 4, pp. 561–599, 2014. DOI: 10.1080/10508406.2014.954750. eprint: https://doi.org/10.1080/10508406.2014.954750. [Online]. Available: https://doi.org/10.1080/10508406.2014.954750.

[21] E. Lyulina, A. Birillo, V. Kovalenko, and T. Bryksin, "Tasktracker-tool: A toolkit for tracking of code snapshots and activity data during solution of programming tasks," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '21, ACM, Mar. 2021. DOI: 10.1145/3408877.3432534. [Online]. Available: http://dx.doi.org/10.1145/3408877.3432534.

[22] M. Karam, M. Awa, A. Carbone, and J. Dargham, "Assisting students with typical programming errors during a coding session," in *2010 Seventh International Conference on Information Technology: New Generations*, 2010, pp. 42–47. DOI: 10.1109/ITNG.2010.244.

[23] A. Ahadi, R. Lister, and L. Mathieson, "Aral: An online tool for source code snapshot metadata analysis," in *Proceedings of the Twenty-First Australasian Computing Education Conference*, ser. ACE '19, Sydney, NSW, Australia: Association for Computing Machinery, 2019, 118–125, ISBN: 9781450366229. DOI: 10.1145/3286960.3286975. [Online]. Available: https://doi.org/10.1145/3286960.3286975.

[24] H. Crompton and D. Burke, "Artificial intelligence in higher education: the state of the field," *Int. J. Educ. Technol. High. Educ.*, vol. 20, no. 1, pp. 1–22, Dec. 2023, ISSN: 2365-9440. DOI: 10.1186/s41239-023-00392-8.

[25] Y. Choi and C. McClenen, "Development of adaptive formative assessment system using computerized adaptive testing and dynamic bayesian networks," *Applied Sciences*, vol. 10, no. 22, 2020, ISSN: 2076-3417. DOI: 10.3390/app10228196. [Online]. Available: https://www.mdpi.com/2076-3417/10/22/8196.

[26] L. Chen, P. Chen, and Z. Lin, "Artificial intelligence in education: A review," *IEEE Access*, vol. 8, pp. 75 264–75 278, 2020. DOI: 10.1109/ACCESS.2020.2988510.

[27] S. Pokrivcakova, "Preparing teachers for the application of AI-powered technologies in foreign language education," *Journal of Language and Cultural Education*, vol. 7, no. 3, pp. 135–153, Dec. 2019. DOI: 10.2478/jolace-2019-0025.

[28] O. Hamal, N.-E. El Faddouli, M. H. A. Harouni, and J. Lu, "Artificial intelligent in education," *Sustainability*, vol. 14, no. 5, 2022, ISSN: 2071-1050. DOI: 10.3390/su14052862. [Online]. Available: https://www.mdpi.com/2071-1050/14/5/2862.

[29] P. Rivas, D. R. Schwartz, and E. Quevedo, "Bert goes to sql school: Improving automatic grading of sql statements," in *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, 2023, pp. 83–90. DOI: 10.1109/CSCE60160.2023.00019.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1810.04805.

[31] J. Wang, Y. Zhao, Z. Tang, and Z. Xing, "Combining dynamic and static analysis for automated grading sql statements," *J Netw Intell*, vol. 5, no. 4, pp. 179–190, 2020.

[32] P. Rivas and D. R. Schwartz, "Modeling sql statement correctness with attention-based convolutional neural networks," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2021, pp. 64–71. DOI: 10.1109/CSCI54926.2021.00086.

[33] *Captum*, [Online; accessed 16. Jun. 2024], Jun. 2024. [Online]. Available: https://github.com/pytorch/captum.

[34] *OpenAI Platform*, [Online; accessed 16. Jun. 2024], Jun. 2024. [Online]. Available: https://platform.openai.com/docs/guides/text-generation.

[35] OpenAI, J. Achiam, S. Adler, *et al.*, *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL].

[36] L. Yan, L. Sha, L. Zhao, *et al.*, "Practical and ethical challenges of large language models in education: A systematic scoping review," *British Journal of Educational Technology*, vol. 55, no. 1, 90–112, Aug. 2023, ISSN: 1467-8535. DOI: 10.1111/bjet.13370. [Online]. Available: http://dx.doi.org/10.1111/bjet.13370.

[37] G. Pinto, I. Cardoso-Pereira, D. M. Ribeiro, D. Lucena, A. de Souza, and K. Gama, *Large language models for education: Grading open-ended questions using chatgpt*, 2023. arXiv: 2307.16696 [cs.SE].

[38] G. Kortemeyer, "Toward ai grading of student problem solutions in introductory physics: A feasibility study," *Physical Review Physics Education Research*, vol. 19, no. 2, Nov. 2023. DOI: 10.1103/physrevphyseducres.19.020163.

[39] A. B. Altamimi, "Effectiveness of chatgpt in essay autograding," in *2023 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, 2023, pp. 102–106. DOI: 10.1109/iCCECE59400.2023.10238541.

[40] B. Naismith, P. Mulcaire, and J. Burstein, "Automated evaluation of written discourse coherence using gpt-4," Jan. 2023, pp. 394–403. DOI: 10.18653/v1/2023.bea-1.32.

[41] S. Tobler, "Smart grading: A generative ai-based tool for knowledge-grounded answer evaluation in educational assessments," *MethodsX*, vol. 12, p. 102 531, Jun. 2024. DOI: 10.1016/j.mex.2023.102531.

[42] I. Azaiz, N. Kiesler, and S. Strickroth, *Feedback-generation for programming exercises with gpt-4*, 2024. arXiv: 2403.04449 [cs.AI].

[43] W. Dai, J. Lin, H. Jin, *et al.*, "Can large language models provide feedback to students? a case study on chatgpt," in *2023 IEEE International Conference on Advanced Learning Technologies (ICALT)*, 2023, pp. 323–325. DOI: 10.1109/ICALT58122.2023.00100.

[44] H. Touvron, T. Lavril, G. Izacard, *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[45] T. Phung, V.-A. Pădurean, A. Singh, *et al.*, "Automating human tutor-style programming feedback: Leveraging gpt-4 tutor model for hint generation and gpt-3.5 student model for hint validation," in *Proceedings of the 14th Learning Analytics and Knowledge Conference*, ser. LAK '24, Kyoto, Japan: Association for Computing Machinery, 2024, 12–23, ISBN: 9798400716188. DOI: 10.1145/3636555.3636846. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/3636555.3636846.

[46] J. Wei, X. Wang, D. Schuurmans, *et al.*, *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL].

[47] T. Phung, J. Cambronero, S. Gulwani, *et al.*, *Generating high-precision feedback for programming syntax errors using large language models*, 2023. arXiv: 2302.04662 [cs.PL].

[48] *OpenAI Codex*, [Online; accessed 12. May 2024], May 2024. [Online]. Available: https://openai.com/index/openai-codex.

[49] M. Pankiewicz and R. S. Baker, *Large language models (gpt) for automating feedback on programming assignments*, 2023. arXiv: 2307.00150 [cs.HC].

[50] L. Yan, R. Martinez-Maldonado, and D. Gasevic, "Generative artificial intelligence in learning analytics: Contextualising opportunities and challenges through the learning analytics cycle," in *Proceedings of the 14th Learning Analytics and Knowledge Conference*, ser. LAK '24, Kyoto, Japan: Association for Computing Machinery, 2024, 101–111, ISBN: 9798400716188. DOI: 10.1145/3636555.3636856. [Online]. Available: https://doi.org/10.1145/3636555.3636856.

[51] H. Prematilaka, *Learner Agency: Maximizing Learner Potential | Oxford University Press*, [Online; accessed 12. May 2024], Dec. 2022. [Online]. Available: https://elt.oup.com/feature/global/expert/learner-agency?cc=nl&selLanguage=en.

[52] J. K. Matelsky, F. Parodi, T. Liu, R. D. Lange, and K. P. Kording, *A large language model-assisted education tool to provide feedback on open-ended responses*, 2023. arXiv: 2308.02439 [cs.CY].

[53] *OpenAI Platform*, [Online; accessed 22. May 2024], May 2024. [Online]. Available: https://platform.openai.com/docs/guides/prompt-engineering.

[54] L. Weng, "Prompt engineering," *lilianweng.github.io*, Mar. 2023. [Online]. Available: https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/.

[55] *Prompt Engineering Guide | Prompt Engineering Guide*, [Online; accessed 23. May 2024], May 2024. [Online]. Available: https://www.promptingguide.ai.

[56] J. White, Q. Fu, S. Hays, *et al.*, *A prompt pattern catalog to enhance prompt engineering with chatgpt*, 2023. arXiv: 2302.11382 [cs.SE].

[57] *OpenAI Platform*, [Online; accessed 14. May 2024], May 2024. [Online]. Available: https://platform.openai.com/docs/guides/text-generation/managing-tokens.

[58] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL].

[59] T. Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, *Calibrate before use: Improving few-shot performance of language models*, 2021. arXiv: 2102.09690 [cs.CL].

[60] X. Wang, J. Wei, D. Schuurmans, *et al.*, *Self-consistency improves chain of thought reasoning in language models*, 2023. arXiv: 2203.11171 [cs.CL].

[61] M. T. Ribeiro, S. Singh, and C. Guestrin, *"why should i trust you?": Explaining the predictions of any classifier*, 2016. arXiv: 1602.04938 [cs.LG].

[62] M. Cukurova, R. Luckin, and C. Kent, "Impact of an Artificial Intelligence Research Frame on the Perceived Credibility of Educational Research Evidence," *Int. J. Artif. Intell. Educ.*, vol. 30, no. 2, pp. 205–235, Jun. 2020, ISSN: 1560-4306. DOI: 10.1007/s40593-019-00188-w.

[63] M. T. Dzindolet, S. A. Peterson, R. A. Pomranky, L. G. Pierce, and H. P. Beck, "The role of trust in automation reliance," *International Journal of Human-Computer Studies*, vol. 58, no. 6, pp. 697–718, 2003, Trust and Technology, ISSN: 1071-5819. DOI: https://doi.org/10.1016/S1071-5819(03)00038-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1071581903000387.

[64] langchain ai, *langchain*, [Online; accessed 21. May 2024], May 2024. [Online]. Available: https://github.com/langchain-ai/langchain.

[65] *Hello GPT-4o*, [Online; accessed 19. Jun. 2024], Jun. 2024. [Online]. Available: https://openai.com/index/hello-gpt-4o.

[66] *Streamlit Docs*, [Online; accessed 21. May 2024], May 2024. [Online]. Available: https://docs.streamlit.io.

[67] *SQL Language Expressions*, [Online; accessed 24. Jun. 2024], Jun. 2024. [Online]. Available: https://www.sqlite.org/lang_expr.html#the_like_glob_regexp_and_match_operators.

[68] *EIP*, [Online; accessed 14. Jun. 2024], Feb. 2024. [Online]. Available: https://eip.pages.ewi.tudelft.nl/eip-website/weblab.html.

[69] G. M. Fernandez Nieto, K. Kitto, S. Buckingham Shum, and R. Martinez-Maldonado, "Beyond the learning analytics dashboard: Alternative ways to communicate student data insights combining visualisation, narrative and storytelling," in *LAK22: 12th International Learning Analytics and Knowledge Conference*, ser. LAK22, Online, USA: Association for Computing Machinery, 2022, 219–229, ISBN: 9781450395731. DOI: 10.1145/3506860.3506895. [Online]. Available: https://doi.org/10.1145/3506860.3506895.