Direct Automated Feedback Delivery for Student Submissions based on LLMs

MAXIMILIAN SÖLCH, Technical University of Munich, Germany
FELIX T.J. DIETRICH, Technical University of Munich, Germany
STEPHAN KRUSCHE, Technical University of Munich, Germany

Timely and individualized feedback is crucial for students' learning progress and motivation. However, providing such feedback is a major challenge in education, especially as the number of students has steadily increased in recent years. This growth has made it difficult for teachers to provide individualized feedback to each student, resulting in a time-consuming, repetitive and often manual task that contributes to a high workload.

This paper presents Direct Automated Feedback (DAFeeD), a large language model (LLM)-based approach for automated formative feedback on student submissions in various exercise domains. The defined feedback process enables interactive learning by allowing students to submit their solutions multiple times and request feedback repeatedly before the submission deadline. By incorporating task-specific information into the prompt, DAFeeD provides iterative customized feedback, facilitating continuous student improvement.

To empirically evaluate the feedback process, we implemented it in an open-source reference implementation that is integrated into the LMS learning platform. We conducted a controlled study in which students used the feedback process in a programming task in a supervised environment and completed a survey. The results show that students perceive the feedback process as relevant and beneficial to their learning. Students indicated that they feel more comfortable and willing to request automated feedback than they would with human tutors because it is more convenient and immediate. This shows that DAFeeD has the potential to significantly improve the feedback process in educational institutions and increase students' learning efficiency and performance.

CCS Concepts: • **Social and professional topics** → **Student assessment**; • **Applied computing** → **Education**.

Additional Key Words and Phrases: Software Engineering, Education, Formative Feedback, Interactive Learning

ACM Reference Format:
Maximilian Sölch, Felix T.J. Dietrich, and Stephan Krusche. 2024. Direct Automated Feedback Delivery for Student Submissions based on LLMs. In . ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the current educational landscape, providing timely and effective feedback to students remains a significant challenge. Traditionally, students must wait for course tutors or professors to review their submissions and provide feedback. This process can be time-consuming, often requiring students to arrange meetings and wait for available time slots, which are not always convenient or immediate. Similar it is time-consuming and tedious for professors and tutors to provide asynchronous feedback via email or other communication channels [5]. The inherent delays and scheduling difficulties make this approach not scalable, especially in courses with a large number of students.

These limitations hinder students' learning progress and motivation. The waiting period for feedback interrupts the learning flow, causing students to lose momentum and potentially disengage from the subject matter. Additionally, providing individualized feedback and enabling students to enhance their knowledge through formative assessments

Manuscript submitted to ACM

are crucial components of effective learning [6, 7]. However, the limited availability of tutors and professors means that not all students receive the individualized attention they need to improve their understanding and skills. This situation underscores the necessity for a more efficient and scalable feedback system that can provide continuous support and formative feedback to students without the constraints of traditional methods [16].

In this paper, we present DAFeeD, an approach for generating automated feedback on student submissions using large language models (LLMs), to address these challenges. The approach is independent of the exercise type and can be applied to various domains, such as programming, text, or modeling exercises. We implemented the approach in an open-source reference implementation called Athena, connected to the learning platform Artemis through which students submit their solutions and receive feedback. To validate the effectiveness and efficiency of the approach, we tested it in a controlled environment.

With this paper, we want to answer the following research questions about direct automated feedback delivery:

RQ1 How does the availability of direct automated feedback affect student engagement and motivation?
RQ2 Do students feel more comfortable requesting automatic feedback than asking a human tutor or the course professor?
RQ3 How do students perceive the effectiveness of direct automated feedback?
RQ4 How do students perceive the usability and helpfulness of DAFeeD?

The subsequent sections of this paper are organized to provide a comprehensive understanding of the research. Section 2 provides an overview of related work. Section 3 details the concept and methodology of Direct Automated Feedback Delivery (DAFeeD). Section 4 describes the reference implementation of DAFeeD, called Athena, including a general overview, details on the used prompts, and the system architecture. Section 5 presents the evaluation results, including the research questions, study design, and findings. Finally, Section 6 concludes with a summary of findings and discusses future research directions to enhance automated feedback systems.

2 RELATED WORK

Automated feedback systems have gained significant attention in educational research due to their potential to scale online education and reduce the time between submission and feedback. While Hahn et al. [4] conducted a systematic review on the effects of automatic scoring and feedback tools, emphasizing their crucial role in enhancing scalability, reducing bias, and increasing student engagement. Their insights on these aspects are highly relevant to our study, as they highlight the broader implications of automated feedback systems in education. Additionally, Keuning et al. [8] reviewed 101 tools for automated feedback on programming exercises, noting that most focus on error identification rather than providing actionable guidance or adapting to specific instructional needs. Extending this work, Kiesler et al. [9] explored the effectiveness of large language models (LLMs) like ChatGPT in generating formative programming feedback, finding that while LLMs can produce useful feedback, they often include misleading information for novices.

Shute [15] provided a comprehensive review of formative feedback, highlighting the importance of feedback being nonevaluative, supportive, timely, and specific. Shute emphasized that effective formative feedback should be clear and actionable to improve learning outcomes, aligning with our DAFeeD approach where such feedback is provided iteratively. Similarly, Dawson et al. [3] conducted a qualitative investigation into perceptions of effective feedback, revealing that while educators focus on design aspects such as timing and modalities, students prioritize the quality and usability of feedback comments. This distinction underscores the need for automated feedback systems to deliver not only timely feedback but also detailed, specific, and personalized comments to individual students' work.

Moreover, Marwan et al. [12] demonstrated the importance of adaptive and immediate feedback in enhancing student learning outcomes, showing that such feedback mechanisms can significantly improve student performance and motivation. Similarly, Leinonen et al. [10] compared immediate and scheduled feedback, concluding that immediate feedback is more effective in promoting student engagement and timely corrections. These studies collectively underscore the potential of automated feedback systems in providing timely, adaptive, and engaging feedback to students, which is crucial for their continuous improvement and learning efficiency.

The work by Azaiz et al. [2] highlights the limitations of LLMs, advising against using GPT-4 Turbo for automatic feedback generation for programming education due to inconsistencies. In contrast, our research evaluates the integrated direct automatic feedback delivery process within an LMS, which we believe can already be beneficial for students. We anticipate that increasingly powerful LLMs and advanced prompting strategies will improve the feedback generation process over time without revealing solutions.

The study by Liffiton et al. [11] introduces CodeHelp, an LLM-powered tool that provides real-time assistance to programming students. In a first-year computer science course with 52 students, CodeHelp collected data over 12 weeks, revealing that students valued its availability, immediacy, and support for error resolution and independent learning. CodeHelp requires students to manually enter code, error messages, and issue descriptions. In contrast, DAFeeD integrates into the LMS, automatically providing context and feedback on code repository changes without requiring student input. This seamless integration aims to increase student engagement and motivation by offering timely, relevant feedback with less effort from students, and to improve perceptions of feedback effectiveness, usability, and helpfulness.

Nguyen and Allan [13] demonstrate the feasibility of using GPT-4 for tiered, formative feedback on programming exercises in introductory courses, providing insights on conceptual understanding, syntax, and time complexity. Similarly, our DAFeeD approach uses GPT-4 Turbo and focuses on providing formative feedback in introductory courses. However, while **?** provide feedback on isolated code snippets using few-shot learning, DAFeeD delivers iterative feedback on entire repositories with multiple files using a zero-shot approach with detailed prompts and context collection. Additionally, DAFeeD is integrated directly into an LMS, supporting an exercise-independent feedback process, which allows students to iteratively improve their submissions without revealing solutions.

The work by Woodrow et al. [18] explores the deployment and effectiveness of a real-time style feedback tool using LLMs, specifically GPT 3.5 Turbo, in a large-scale online CS1 course. Their findings indicate significant improvements in student engagement and coding style when feedback is immediate and integrated within the learning platform. **?** conducted a randomized control trial with over 8,000 students, demonstrating that real-time feedback was five times more likely to be viewed and incorporated by students compared to delayed feedback. This supports our approach with DAFeeD, emphasizing the importance of immediate, iterative, and context-specific feedback in enhancing student learning outcomes.

3 APPROACH: DIRECT AUTOMATED FEEDBACK DELIVERY (DAFEED)

DAFeeD employs large language models to deliver automated feedback on student submissions, designed to complement traditional teaching methods and provide additional support. Figure 1 illustrates the continuous feedback workflow that DAFeeD facilitates, enabling students to receive feedback at any time, thereby eliminating the need to wait for responses from human tutors or course professors.

The feedback process is designed to be exercise-independent, meaning that it can be applied to various exercise types, such as programming, text, or modeling exercises. DAFeeD can provide formative feedback to the students, including

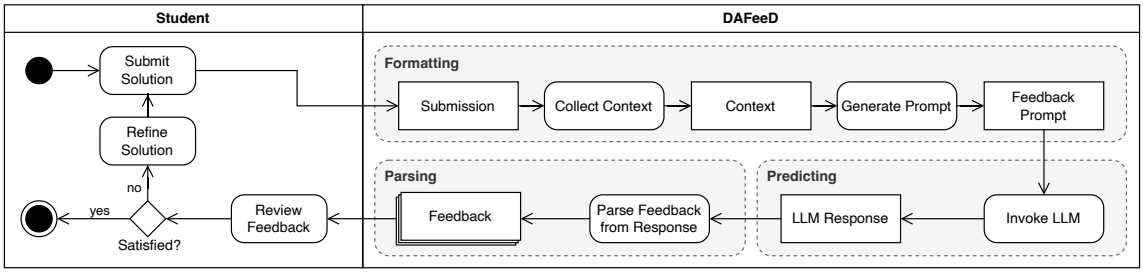

Fig. 1. Workflow of direct automated feedback delivery (DAFeeD) for students' submissions (UML Activity Diagram)

feedback on issues or improvements, as well as positive feedback when the student completes the task correctly. Once the student submits their solution, DAFeeD initiates a three-stage process to generate natural language feedback.

The first stage, called *Formatting*, takes the student's submission and extracts the submission content, the problem statement, including learning objectives, and any possible grading instructions the instructor defines. All of this gathered information represents the context. During the prompt generation step, a predefined prompt template is filled with the prompt input data, resulting in the feedback prompt. Depending on the exercise, adaptions need to be made to the prompt template to ensure that the feedback output of the LLM is tailored to the specific exercise type. For programming exercises, the generated feedback needs to have metadata information about the file and line number of the code snippet to which the feedback refers. For text exercises, the feedback needs to have metadata about the sentence or word range the feedback refers to.

In the second stage, called *Predicting*, DAFeeD sends the feedback prompt to a large language model (LLM) and invokes it with the prompt. As a result, the LLM generates a response to that prompt including detailed feedback items for the student.

The final stage, *Parsing*, takes the LLM response, which comes in the JSON format, and parses feedback items from it. In addition to the feedback text, the feedback object also contains reference information indicating the part of the submission it pertains to. For programming exercises, this includes the file name and line number of the relevant code snippet to which the feedback refers. For text exercises, the reference information includes only the sentence or word range the feedback refers to.

All of the feedback is then returned to the student for review. If the student is satisfied with the feedback, the process concludes. Otherwise, the student can refine and resubmit their solution, initiating the DAFeeD process anew.

This iterative process is designed to motivate students to continuously learn and experiment with their solutions, resulting in improved performance.

4 REFERENCE IMPLEMENTATION: ATHENA

We incorporated DAFeeD into a reference implementation named Athena, which is seamlessly integrated with the learning platform Artemis. Through Artemis, students can submit their solution and review the feedback.

When submitting their solutions on Artemis, students have the option to request direct automated feedback by clicking a newly added button. This feedback request is then sent to Athena, provided the student has not reached their feedback request limit for the exercise. Course instructors can customize the number of allowed feedback requests per exercise according to their preference. A status visualization informs students about their feedback request state. Once

Athena generates the feedback and sends it back to Artemis, the student can review it in a modal window on Artemis, as depicted in Figure 2.

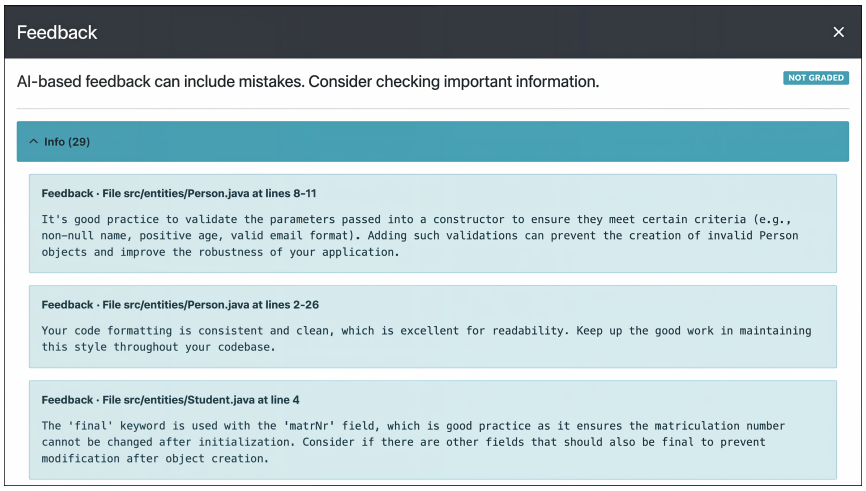

Fig. 2. Visualization of the feedback how students see it in Artemis.

4.1 Feedback Generation

The prompt design is crucial for guiding the large language model (LLM) in generating effective and contextually relevant feedback. In Figure 3, we provide an example of a prompt used for generating feedback for programming exercises. This prompt incorporates specific instructions to ensure that the feedback is tailored to the student's submission.

The feedback generation process begins by identifying the differences between the student's submission repository and the provided template repository, the starting point. These differences are identified using a git diff, which highlights lines removed and added by the student. If the problem statement is too lengthy or complex, a separate LLM invocation is used to split the problem statement into relevant parts for each file. This ensures that the feedback is targeted and relevant to the specific context of the file being reviewed. Additionally, a summary of the student's solution across all files is generated using another LLM invocation. This summary provides a comprehensive overview of the submission, which is included in the prompt to offer context for the feedback.

In the provided prompt, several key components guide the AI in creating useful feedback. The *Problem Statement* section contextualizes the student's task and helps the AI understand the exercise's objectives. The *Task Instructions* direct the AI to provide non-graded, constructive feedback focusing on educational aspects without offering direct solutions. *Style Guidelines* ensure the feedback is constructive, specific, balanced, clear, concise, actionable, educational, and contextual. The *File Path and Content* provide the specific file under review along with its content, aiding the AI in pinpointing specific lines of code for feedback. Additionally, *Summary and Diffs* between the template and submission offer additional context, helping the AI understand the student's changes and their overall approach.

The structure and content of this prompt are designed to emulate a human tutor's approach, ensuring that the feedback is both relevant and supportive of the student's learning process. By providing such detailed instructions and contextual information, the LLM can generate feedback that is both meaningful and actionable for students.

```
You are an AI tutor for programming assessment at a prestigious university.

# Problem statement
{problem_statement}

# Task
Provide constructive, non-graded feedback on a student's programming submission as a
human tutor would. The tutor is not familiar with the solution, so the feedback should
focus solely on aspects from which the student can learn. This feedback must highlight
incorrectly applied principles or inconsistencies without offering specific solutions or
error corrections. Allow some flexibility for students to deviate from the problem
statement, provided they complete all tasks. Ensure the feedback is balanced and
comprehensive.

# Style
1. Constructive, 2. Specific, 3. Balanced, 4. Clear and Concise, 5. Actionable, 6.
Educational, 7. Contextual

Feedback that contradicts the problem statement is strictly prohibited. Avoid mentioning
aspects not explicitly covered in the template to submission diff, such as the exercise
package name, as these are beyond the student's control.

In git diff, lines marked with '-' were removed and with '+' were added by the student.

The student will be reading your response, so use "you" instead of "them".

Path: {submission_file_path}

File (with line numbers <number>: <line>):
{submission_file_content}

Summary of other files in the solution:
{summary}

The template to submission diff (only as reference):
{template_to_submission_diff}
```

Fig. 3. Prompt for Generating Feedback for Programming Exercises. The highlighted sections are placeholders for the respective elements.

4.2 Architecture

Athena is deployed in production alongside the learning platform Artemis, which serves up to more than 2000 students per course. Consequently, the reference implementation must satisfy additional non-functional requirements such as performance, scalability, maintainability, and usability. To meet these requirements and to support feedback generation for multiple exercise types while allowing for future extensibility, we adopted a modular architecture, as illustrated in Figure 4.

The *Module Manager* handles all incoming requests, verifies authorization, and forwards them to the appropriate modules. The *ProgrammingLLM* module manages programming exercises and executes the three-stage DAFeeD process, which includes formatting, predicting, and parsing. Similarly, the *TextLLM* module is optimized for text exercises and follows the same process.

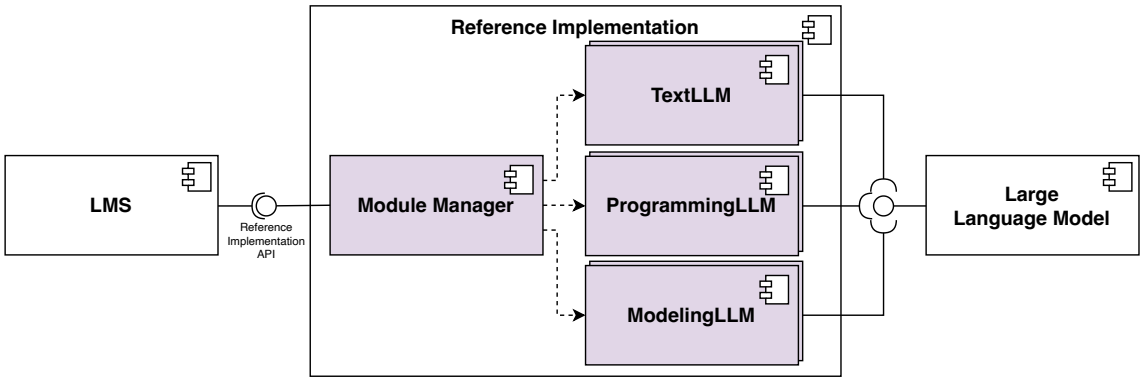

Fig. 4. Top Level Archietecture of the reference implementation Athena (UML Deployment Diagram)

Athena's system design is independent of any specific learning management system (LMS) as it provides a REST API, documented using the OpenAPI standard[1]. This independence allows Athena to be integrated with various LMS platforms, such as Moodle[2].

Athena currently connects to OpenAI models hosted in a private Azure cloud to ensure that student data is not used for training models, maintaining privacy. Additionally, the system can be configured to use open-source models like Llama[3] or Mistral[4], either self-hosted or cloud-based.

To meet performance and scalability requirements, Athena and its modules are deployed within a Kubernetes cluster[5]. Kubernetes, in conjunction with Athena's modular architecture, allows the system to scale each module independently. For example, additional instances of the programming module can be instantiated when a new programming exercise is released. Furthermore, Kubernetes provides out-of-the-box load balancing and self-healing capabilities, ensuring that if a module crashes, it is automatically restarted.

5 EVALUATION

In this section, we outline the methodology employed to validate the effectiveness of the proposed DAFeeD approach including the reference implementation Athena. The conducted evaluation represents the treatment validation stage of the design science methodology proposed by Wieringa [17]. In this stage, the proposed solution — DAFeeD — is evaluated in a controlled environment, and the collected data is utilized for the refinement and improvement of the solution.

We begin by describing the study design and the results. Subsequently, we outline the limitations of the evaluation and discuss the implications of the findings.

5.1 Study Design

To gain comprehensive insights into students' perceptions of the newly introduced DAFeeD concept, we employed a survey-based approach, visualized in Figure 5. At the beginning of the study, participants received a two-page instruction

[1] https://www.openapis.org
[2] https://moodle.org
[3] https://llama.meta.com
[4] https://mistral.ai
[5] https://kubernetes.io

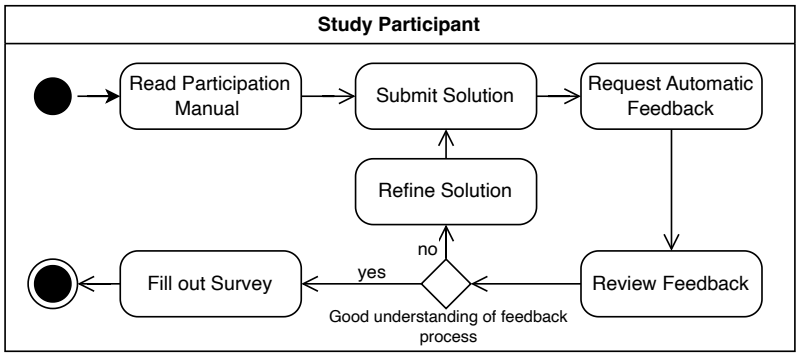

Fig. 5. An UML Activity Diagram showcasing the design and procedure of the conducted study from a participant's perspectives.

manual that included all required informaton. Initially, study participants tested the new feedback feature on a sample exercise using the Artemis platform in a controlled environment at the university.

The time limit for the evalution was set to 45 minutes, including the testing phase of the feature and the succeeding survey. Goal of the hands on phase was to understand the new feedback process, completing the exercise was not required.

Following this hands-on experience, participants were asked to complete a survey hosted on the community version of the open-source survey tool LimeSurvey[6]. This survey aimed to gather their opinions on direct automated feedback and collect feedback on their overall experience with the feature.

We invited students from current courses at the university to participate in the study via direct messages and conducted the study with a total of 20 participants. The study uses a mixed methods approach, combining quantitative and qualitative data collection methods. The participants were a mix of undergraduate and graduate students from various disciplines, including computer science, information systems, and games engineering.

All survey questions, except for the introductory demographic queries and the five final, voluntary free-text responses (Q18 - Q22), employ a 5-point Likert scale [1] ranging from "strongly agree" to "strongly disagree" and are mandatory. The following list shows the mapping from the asked questions and statements to the research questions:

RQ1 Engagement and Motivation

- **Q1** The direct automated feedback keeps me more engaged in the learning process.
- **Q2** The direct automatic feedback motivates me to repeatedly improve my code.
- **Q3** The direct automated feedback makes me feel more motivated to complete my programming assignments.
- **Q4** The direct automated feedback encourages me to experiment more with my coding solutions.

RQ2 Comfort with Feedback Source

- **Q5** I feel more comfortable requesting direct automated feedback than feedback from a human tutor.
- **Q6** I am likely to request feedback more frequently when using direct automated feedback than feedback from my course professor.
- **Q7** I find receiving direct automated feedback less intimidating than receiving feedback from a human tutor.
- **Q8** I feel that requesting direct automated feedback is more convenient than arranging a meeting with a human tutor.

[6] https://www.limesurvey.org

RQ3 Perceived Effectiveness

Q9 The direct automated feedback helps me understand my mistakes.

Q10 The direct automated feedback is more effective than one-time feedback.

Q11 The direct automated feedback has significantly improved the quality of my programming assignment.

Q12 The direct automated feedback is a helpful addition to the automatic test case results.

Q13 I feel that having access to direct automated feedback continuously helps me more than arranging a meeting with a human tutor.

RQ4 Usability and Helpfulness

Q14 It is easy to receive direct automated feedback on my programming assignments.

Q15 I would rather use the direct automated feedback integrated into Artemis than use an external AI tool for getting feedback.

Q16 I find the direct automated feedback helpful in improving my programming skills.

Q17 I am satisfied with the overall performance of the direct automated feedback.

Q18 Are there any improvements that you would suggest for direct automated feedback?

Q19 How did you find the feedback?

Q20 What kind of feedback would you like to receive?

Q21 Was there anything you particularly liked about the direct automated feedback process?

Q22 What difficulties did you encounter when using the direct automated feedback process?

5.2 Results

In the following paragraphs, we present the results. The answers to each of the likert scale questions are visualized in Figure 6.

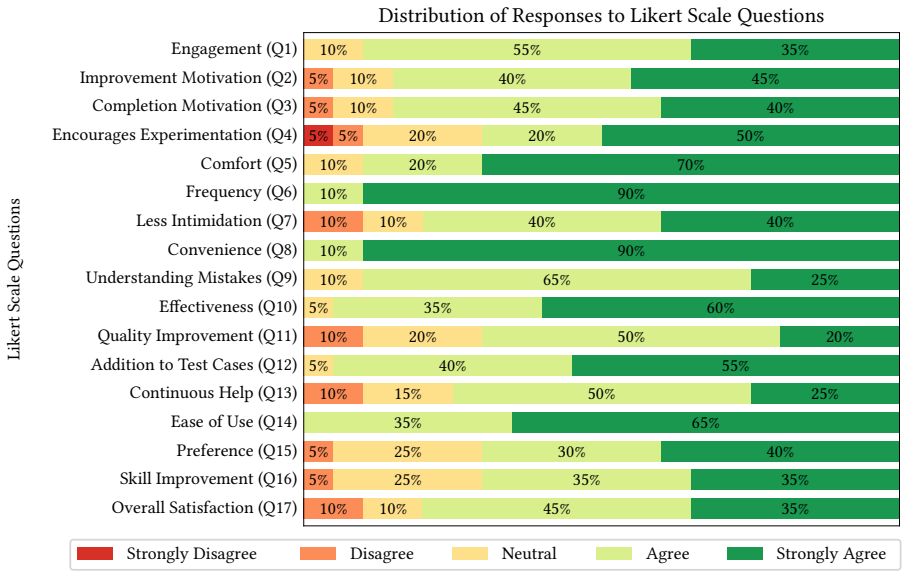

Fig. 6. Distribution of Likert scale responses.

90% of students indicated that the direct automated feedback from Artemis keeps them more engaged in the learning process, with 10% neutral (Q1). For Q2, 85% of students stated that the direct automatic feedback from Artemis motivates them to repeatedly improve their code, with 10% neutral and 5% disagreeing. For Q3, 85% of students mentioned that the direct automated feedback from Artemis makes them feel more motivated to complete their programming assignments, with 10% neutral and 5% disagreeing. The encouragement to experiment more with coding solutions (Q4) received a positive response, with 70% agreeing, 20% neutral, and 10% disagreeing.

Comfort levels in requesting feedback showed that 90% of students feel more comfortable requesting direct automated feedback from Artemis than feedback from a human tutor, with 10% neutral (Q5). For Q6, 100% of students noted that they are likely to request feedback more frequently when using direct automated feedback from Artemis than feedback from their course professor. Receiving feedback from Artemis was found to be less intimidating than from a human tutor by 80% of students, with 10% neutral and 10% disagreeing (Q7). Convenience in requesting feedback showed that 100% of students feel that requesting direct automated feedback from Artemis is more convenient than arranging a meeting with a human tutor (Q8).

In terms of understanding mistakes, 90% of students believed that the direct automated feedback provided by Artemis helps them understand their mistakes, with 10% neutral (Q9). The effectiveness of the feedback was highlighted by 95% of students who found that the direct automated feedback from Artemis is more effective than one-time feedback, with 5% neutral (Q10). Regarding the quality of assignments, 70% of students observed that the direct automated feedback has significantly improved the quality of their programming assignments, with 20% neutral and 10% disagreeing (Q11). For Q12, 95% of students felt that the direct automated feedback is a helpful addition to the automatic test case results, with 5% neutral. Continuous access to feedback was found to be more beneficial than arranging meetings with a tutor by 75% of students, with 15% neutral and 10% disagreeing (Q13).

Ease of receiving feedback was highly rated, with 100% of students confirming that it is easy to receive direct automated feedback from Artemis on their programming assignments (Q14). Furthermore, 70% of students preferred using the direct automated feedback integrated into Artemis over using an external AI tool for getting feedback, with 25% neutral and 5% disagreeing (Q15). In terms of skill improvement, 70% of students agreed that they find the direct automated feedback from Artemis helpful in improving their programming skills, with 25% neutral and 5% disagreeing (Q16). Lastly, 80% of students were satisfied with the overall performance of the direct automated feedback system, with 10% neutral and 10% disagreeing (Q17).

The responses to the voluntary free text questions highlight several themes. Many students appreciated the immediate availability of feedback, which allowed for prompt corrections without waiting for manual review. However, some respondents suggested improvements such as better categorization of feedback, more detailed explanations of errors, and prioritization of critical issues. The feedback was generally found to be relevant and useful in addressing obvious mistakes and improving code quality. Students expressed a preference for feedback that clearly identified mistakes and provided specific guidance on how to correct them, along with suggestions for improvement. Some challenges included understanding certain automated feedback messages and occasional false positives or negatives in error detection.

5.3 Findings

The responses to RQ1 show that the availability of direct automated feedback can significantly enhance student engagement and motivation. Students reported feeling more engaged in the learning process. The majority of participants also stated they are motivated to repeatedly improve their code and complete their programming assignments. Additionally, the feedback encouraged a notable percentage of participants to experiment more with their coding solutions. These

findings suggest that direct automated feedback is highly effective in boosting both engagement and motivation among students.

> **Main Findings for RQ1:** The availability of direct automated feedback significantly enhances student engagement and motivation. Students feel more engaged in the learning process, motivated to improve their code, and encouraged to experiment more with their coding solutions, without having to wait for manual feedback.

The responses to RQ2 reveal a strong level of comfort for requesting automated feedback compared to traditional human feedback channels. Students stated they feel more comfortable requesting automated feedback than from human tutors and were likely to request automated feedback more frequently than from their course professors. Requesting automated feedback was perceived as less intimidating for most of the participants and all of them stated it is more convenient than arranging meetings with a human tutor. These findings highlight the effectiveness of automated feedback in providing a more comfortable and accessible feedback mechanism for students.

> **Main Findings for RQ2:** Students feel more comfortable requesting automated feedback than from human. They are likely to request automated feedback more frequently and find it less intimidating and more convenient than arranging meetings with a human tutor.

The responses to RQ3 indicate that students think automated feedback is highly effective in helping them understand and improve their programming assignments. Students reported that the feedback helped them understand their mistakes and found it more effective than receiving only one-time feedback for their submission. The majority reported that the feedback significantly improved the quality of their programming assignments, and all participants stated that automatic feedback is a helpful addition to automatic test case results generated by Artemis. In addition, most participants saw continuous access to autamted feedback as more beneficial than arranging meetings with a tutor. These findings suggest that automated feedback not only aids in error identification but also significantly enhances the overall quality of student assignments.

> **Main Findings for RQ3:** Students perceive automated feedback as highly effective in helping them understand and improve their programming assignments. The feedback helps them understand their mistakes, improves the quality of their assignments, and is a helpful addition to automatic test case results.

The responses to RQ4 demonstrate the ease of receiving feedback and overall satisfaction with DAFeeD's feedback process and its reference implementation. Students found it easy to receive feedback on their programming assignments. A large number of participants preferred using the feedback integrated into Artemis than copy their submission and relevant context information over to an external AI tool. Most participants also deemed the feedback helpful in improving their programming skills. In regards to the overall performance of the direct automated feedback, the majority of students expressed high satisfaction with the system.

> **Main Findings for RQ4:** Students find it easy to receive feedback on their programming assignments and are satisfied with the overall performance of DAFeeD and its reference implementation. There are some suggestions for improvements, such as better categorization of feedback, more detailed explanations of errors, and prioritization of critical issues.

5.4 Discussion

Overall, the direct automated feedback system was positively received by students, indicating its effectiveness in enhancing their programming skills, despite some areas for improvement.

5.5 Limitations

We follow the categorization framework proposed by Runeson and Höst [14] to outline the limitations of the conducted evaluation, addressing potential threats to internal, external, and construct validity:

Internal Validity: This study may be compromised by using self-reported survey data, which can introduce biases. Participants' perceptions may be influenced by their individual attitudes or varying levels of familiarity with programming concepts, leading to potential inaccuracies. Additionally, the participants' perceived effectiveness does not necessarily correspond to objective effectiveness.

External Validity: Threats to external validity arise from the specific context of this study. Conducting the research exclusively at a single university and with students from computer science, information systems, and similar programs restricts the diversity of the sample. This narrow focus may limit the applicability of the findings to other educational settings or student populations. In addition, the small sample size may also limit the generalizability of the findings.

Construct Validity: The survey questions designed to evaluate 'perceived effectiveness' and 'comfort with feedback source' may not fully encompass the breadth of these constructs. Factors such as prior experiences and personal preferences, which the survey does not account for, could influence participants' responses and perceptions. The use of a specific sample exercise to evaluate the automated feedback process may align more closely with some students' prior experiences or learning styles, introducing bias into the results.

6 CONCLUSION & FUTURE WORK

The main contributions of this paper are the introduction of DAFeeD, a direct automated feedback delivery system, and the reference implementation Athena, which integrates DAFeeD into the learning platform Artemis. DAFeeD enables the interactive learning process by providing students with immediate, context-specific feedback on their submissions. The evaluation of DAFeeD showed that students perceive the automated feedback as effective, helpful, and easy to use, and that it enhances their engagement and motivation.

Future work includes enhancing the visualization of feedback, such as grouping and color coding the feedback items to make it easier to differentiate between critical feedback items, suggestions for improvement, and positive feedback. A high priority will be on further improving the overall quality of the feedback provided. We also aim to extend the implementation to support direct automated feedback for the remaining exercise types of Artemis. Another crucial step is to test direct automated feedback in a real-world setting by utilizing this feature in an actual course. This will allow us to collect comprehensive data to thoroughly evaluate the impact on student performance and motivation.

REFERENCES

[1] I Elaine Allen and Christopher A Seaman. 2007. Likert Scales and Data Analyses. *Quality progress* 40, 7 (2007), 64–65.

[2] Imen Azaiz, Natalie Kiesler, and Sven Strickroth. 2024. Feedback-Generation for Programming Exercises With GPT-4. arXiv:2403.04449 [cs]

[3] Phillip Dawson, Michael Henderson, Paige Mahoney, Michael Phillips, Tracii Ryan, David Boud, and Elizabeth Molloy. 2019. What Makes for Effective Feedback: Staff and Student Perspectives. *Assessment & Evaluation in Higher Education* 44, 1 (Jan. 2019), 25–36. https://doi.org/10.1080/02602938.2018.1467877

[4] Marcelo Guerra Hahn, Silvia Margarita Baldiris Navarro, Luis De La Fuente Valentin, and Daniel Burgos. 2021. A Systematic Review of the Effects of Automatic Scoring and Automatic Feedback in Educational Settings. *IEEE Access* 9 (2021), 108190–108198. https://doi.org/10.1109/ACCESS.2021.3100890

[5] Michael Henderson, Tracii Ryan, and Michael Phillips. 2019. The Challenges of Feedback in Higher Education. *Assessment & Evaluation in Higher Education* 44, 8 (Nov. 2019), 1237–1252. https://doi.org/10.1080/02602938.2019.1599815

[6] Richard Higgins, Peter Hartley, and Alan Skelton. 2002. The Conscientious Consumer: Reconsidering the Role of Assessment Feedback in Student Learning. *Studies in Higher Education* 27, 1 (Feb. 2002), 53–64. https://doi.org/10.1080/03075070120099368

[7] Alastair Irons. 2007. *Enhancing Learning through Formative Assessment and Feedback*. Routledge, London. https://doi.org/10.4324/9780203934333

[8] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education* 19, 1, Article 3 (Sept. 2018). https://doi.org/10.1145/3231711

[9] Natalie Kiesler, Dominic Lohr, and Hieke Keuning. 2023. Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. arXiv:2309.00029 [cs]

[10] Juho Leinonen, Paul Denny, and Jacqueline Whalley. 2022. A Comparison of Immediate and Scheduled Feedback in Introductory Programming Projects. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1 (SIGCSE 2022, Vol. 1)*. Association for Computing Machinery, New York, NY, USA, 885–891. https://doi.org/10.1145/3478431.3499372

[11] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2024. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research (Koli Calling '23)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3631802.3631830

[12] Samiha Marwan, Ge Gao, Susan Fisk, Thomas W. Price, and Tiffany Barnes. 2020. Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. In *Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER '20)*. Association for Computing Machinery, New York, NY, USA, 194–203. https://doi.org/10.1145/3372782.3406264

[13] Ha Nguyen and Vicki Allan. 2024. Using GPT-4 to Provide Tiered, Formative Code Feedback. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 958–964. https://doi.org/10.1145/3626252.3630960

[14] Per Runeson and Martin Höst. 2009. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering* 14, 2 (April 2009), 131–164. https://doi.org/10.1007/s10664-008-9102-8

[15] Valerie J. Shute. 2008. Focus on Formative Feedback. *Review of Educational Research* 78, 1 (March 2008), 153–189. https://doi.org/10.3102/0034654307313795

[16] H. Sondergaard and D. Thomas. 2004. Effective Feedback to Small and Large Classes. In *34th Annual Frontiers in Education, 2004. FIE 2004.* IEEE, Savannah, GA, USA, 540–545. https://doi.org/10.1109/FIE.2004.1408573

[17] Roel J. Wieringa. 2014. *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin Heidelberg.

[18] Juliette Woodrow, Ali Malik, and Chris Piech. 2024. AI Teaches the Art of Elegant Coding: Timely, Fair, and Helpful Style Feedback in a Global Course. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 1442–1448. https://doi.org/10.1145/3626252.3630773