Universiteit Leiden

Master Computer Science

The Effectivity of AI-based Personalized Feedback of Error Message in Programming Education for Higher Education Students

Name: Arief Rahman
Student ID: s4134230
Date: 26/02/2026

Specialisation: Data Science

1st supervisor: Dr. A.N. van der Meulen
2nd supervisor: Dr. Giulio Barbero

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Programming error messages are often difficult for novice programmers to understand, which can hinder the debugging process and negatively affect learning. This study investigates the use of provide personalized feedback on programming error messages and implements artificial intelligence (AI). A web-based application was developed to simulate coding exercises and deliver enhanced error messages using six corrective feedback strategies. A reinforcement learning (RL) model based on the Gradient Bandit method was implemented to adaptively select the most suitable feedback strategy, while a large language model (LLM) was used to generate improved error messages and assess participants' answers. The study involved 24 higher education students with limited or no programming background.

The results indicate that most participants perceived the AI-based personalized feedback positively, as reflected in the Technology Acceptance Model (TAM) survey results. Log data analysis shows that the RL model was able to adaptively recommend feedback strategies, with Explicit Correction being the most frequently selected strategy. The LLM successfully generated enhanced error messages, although its answer assessment performance was sensitive to prompt design. Overall, the findings suggest that AI-based personalized feedback for programming error messages is technically feasible and generally acceptable for novice learners, while further improvements are needed to enhance reliability and scalability.

Contents

1 Introduction

Recent advances in artificial intelligence (AI) have influenced the popularity of the computer science field. The growing interest in computer science, results in the rising number of higher education students in computer science and related disciplines. In Europe, most countries have experienced an increase in the enrollments of Bachelor's computer science programmes over the past five years [4]. Furthermore, the demand of learning computer science subjects from other disciplines has also been expanding with more diverse students [8].

Computer science education can provide a lot of valuable and important skills. Programming is one of the core component skills in computer science that has become an important skill for many other disciplines outside of the computer science field. However, learning programming can be challenging, especially for students with little or no computer science background. These difficulties include understanding the program, structure, algorithm, and debugging [16, 15].

One of the most common difficulties in learning programming is the debugging process. In debugging, the methods that provide information to the user (error message) are more effective than immediate interruption of the program [21]. Therefore, error message plays an important role to help identifying and troubleshooting the problem to solve the errors in a code or program.

However, programming error messages are sometimes difficult to understand [38], especially for people who are still new in learning programming. New programmers can struggle and spend a lot of time trying to understand the error messages [3]. This can be caused by the length, structure, and technical terms of the error message [3], and also the readability of error messages by experts, non-experts, and students are different [3].

Programming error messages can be enhanced to better support the novice programmers. Zhou (2021) [38] explored the effectiveness of Enhanced Programming Error Messages (EPEM), and suggested that more research on the role of errors can be explored further, as mistakes and debugging can contribute to learning programming. Therefore, it is important to have clear and helpful error messages, to help novice programmers in learning programming.

On the other hand, it is also important to consider the human capability in processing the information. Providing excessive information in the error message feedback can cause cognitive overload, thereby reducing learners' ability to effectively process and understand the information needed to solve the problem [29]. Meanwhile, an effective feedback of error message can also be different for each student, because each individual may have different prior knowledge, learning style, personalities, and even each error message might also need different treatment, thus need a different feedback strategy.

Recent study by Pomp (2025)[25] analyzed the influence of personalized feedback of error message on coding education, that personalizes different feedback strategies. However, it did not find any clear evidence that the personalized feedback was the reason of the improvement in the learning process. It also had several limitations, such as the low number and variety of the participants, and some technical issues in the experiment. The study can be explored further to better analyze the use of personalized feedback by addressing the limitations, and also by integrating with AI.

AI has been growing rapidly, and applied widely to support many areas, including education in computer science. This study will extend the work of personalized feedback by Pomp [25] with implementing AI and experiment on broader participants. Hence, it will focus on how AI can be used for the personalized feedback and how the learners perceive the AI-based personalized feedback.

1.1 Research Questions

This research will analyze on how the AI-based personalized feedback of error messages can be helpful in learning programming for students in the university. Additionally, this research will also analyze the implementation of AI for the personalized feedback in programming error messages. A reinforcement learning (RL) model will be applied for the personalization technique, and a large language model (LLM) will be used for the enhanced error message generation.

The study will focus on the following research questions:

- How can AI-based personalized feedback of error messages be helpful in programming education for higher education students?

  The sub-questions are as follows:

  - How do students experience and perceive the AI-based personalized feedback in learning programming?
  - How can RL and LLM be used for AI-based personalized feedback on programming error messages?

2 Related Works

2.1 Computer Science Education

Computer science is a fast growing and evolving field. The rapid evolution of computer science has contributed to the emergence of many new fields [9]. As a result, the demands for education in computer science and related subjects are increasing. This growth is not only in the number of students, but also in the diversity, such as different levels of prior knowledge, experience, background field, skills, motivations, gender, and many other factors [8]. Therefore, computer science education needs to be able to reach broader and more diverse students.

The diversity in the knowledge and/or experience is one of the most frequently occurred. Some countries in Europe have introduced Informatics as a separate compulsory subject since primary school [7]. However, a few other countries still have students that do not receive Informatics education in school (because it was optional for the school and the students) [7]. Hence, students have different levels of prior knowledge/experience in computer science while starting in higher education.

Early exposure to computer science provides a lot of benefit. It can support the development of important skills for the future and to adapt in this era of the digital age [32, 9], these skills include analytical, creativity, problem solving, and computational thinking. Moreover, computer science education can be done as a self-contained discipline and also as both interdisciplinary and transdisciplinary way [9]. On the other hand, students with little or no computer science background may encounter difficulties on subjects that require computer science skills, such as programming. This highlights the need for computer education system that can support students with various background, prior knowledge, and experience.

2.2 Programming Skill and Challenges

Programming is one of the fundamental skills in computer science. It has been widely applied in many other disciplines, and has become an increasingly required skill. As a result, the demand of programming education has also been increasing. However, learning programming can be very challenging, especially for students with little or no background in computer science. Programming is often considered as a cognitively difficult subject by beginners because it involves learning complex new knowledge, strategies, and technical skills [36].

There can be many difficulties in learning programming. These difficulties include understanding the program, structure, algorithm, and debugging [16, 15]. Qian et al. (2017) [26] classifies these difficulties into 3 knowledge misconceptions: syntactic, conceptual, and strategic. Syntactic knowledge misconception is a difficulty in understanding programming syntax. Conceptual knowledge misconception is a difficulty in understanding the programming concept, such as variables, data types, conditions, and many other concepts. Strategic knowledge is a skill about planning, processing, and troubleshooting program. These misconceptions and challenges are mostly related to the student's prior knowledge, and identifying and solving the student's misconceptions is important in programming education [26].

One of the most common difficulties in learning programming is debugging the program. Debugging is an important process to solve problems that occurred in the code/program. Previous studies have explored on what types of knowledge are helpful, and what strategies are used in debugging. These types of knowledge include general programming skill, knowledge of the program, knowledge of the error, and the troubleshooting method [21]. The strategies include determining the problem, understanding the program, exploring the process, evaluating the program, and fixing the error [21, 35]. However, some of the current debugging processes are still ineffective in helping to understand the problems. Beginners often struggle to understand error message, locate the error, and solve the problem effectively. Even people with more programming experience may still have difficulty when debugging program [15].

In debugging process, the methods that provide information to the user (error message) are more effective than immediate interruption of the program [21]. Error messages are a type of feedback that the user receive in the process of debugging a code or program. Programmers rely on the error message feedback, to understand and fix the code [6]. Therefore, error messages play an important role to help identifying and solving errors in a program.

2.3 Programming Error Message

Programming error message contains a set of information that can be used for understanding, locating, and solving the error in the program. However, the information provided in the error messages can be difficult to understand. This could be due to the length, structure, and technical terms of the error message [3]. The raw and obscure error messages are also insufficient to guide the user to solve the problem, and causing the user to be pessimistic [6].

```
---------------------------------------------------------
TypeError                                 Traceback (most
Cell In[8], line 3
      1 x = 10
----> 3 print("There are "+ x +" students.")

TypeError: can only concatenate str (not "int") to str
```

Figure 1: A simple error message.

Figure 1 shows an example of error message that can be hard to understand by new programmers, because it is short and contains some technical terms (data types) that might not have been learnt by the students. The difficulty in understanding the error message has become one of the major barrier that the new programmers encounter [6]. Previous studies have explored on enhancing the error message to better support programmers in solving the error effectively. Becker (2016) [2] designed and implemented enhanced error messages and reported a reduced number of overall errors, however, the enhanced error message is applied manually on each possible error message, thus not reliable and not flexible in the future. Several studies attempted using LLMs to generate enhanced error messages [18, 27, 19] for a more flexible method, however, the results are still not

optimal due to several limitations, such as the model performance, dataset, and prompt design. Nonetheless, these can still be explored to better enhance the error messages.

Designing error message with understandable sentence, avoid difficult or technical terms [12], describing the causes, and providing possible solutions can help students in solving the errors [6], and also help them in learning programming [38]. However, it is also important to consider the human capability in processing the information. Providing excessive information can cause cognitive overload, thereby reducing learners' ability to effectively process and understand the information needed to solve the problem [29].

2.4 Feedback Strategies

In order to learn effectively, the information (from the error messages) presented should not exceed the capacity of human's working memory that process information [29, 22]. The cognitive load theory by John Sweller can be applied to improve the error message feedback, for instance, by managing the intrinsic load by splitting long context of explanation into smaller chunks, reduce the extraneous load by removing technical terms, and enhance the germane load by providing examples.

However, an effective feedback of error message can also be different for each student, because each individual may have different prior knowledge, learning style, personalities, and even each error message might also need different treatment, thus need a personalization. A recent study by Pomp (2025) [25] analyzed the influence of personalized feedback on programming education for children using a list of corrective feedback strategies [33] [30, 1], to find the suitable feedback type for the student. This personalized feedback strategy [25] seems to perform better than using only one feedback strategy, and this study can be further explored, such as by implementing AI methods for both, the feedback strategy recommendation and the error message generation, and experiment on broader participants.

The personalized feedback strategy selection in this study [25] is based on the corrective feedback strategy categories listed by Van Amerongen [33] from Teddick [30] and Astia [1]. Table 1 shows the six categories of corrective feedback strategies.

Explicit correction	The error is clearly mentioned and/or repeated, and then the correction is provided directly.
Recast	The error information is indirectly recreated to the correct answer.
Clarification request	The error is highlighted with an expression and asking for clarification. But no correction is provided.
Metalinguistic clues	The error is indirectly highlighted with expression that contains related information/knowledge.
Elicitation	An expression or statement is formulated to be completed with the correct answer.
Repetition	An expression is repeated by directly highlighting the error.

Table 1: The 6 Corrective Feedback Strategies.

Several studies showed that some of the feedback strategies can perform better than the others [20, 13, 24]. For example, Explicit Correction and Elicitation are more preferred than the other, because these feedback strategies explicitly provide the correct solution.

However, these feedback strategies have their own advantages and disadvantages. For example, feedback strategies that provide explicit answer might be more efficient for user that already has prior knowledge in programming, while the implicit feedback strategies are more effective for user

with no knowledge in programming. Hence, a suitable feedback strategy needs to be selected properly. This can be done using an AI model that can learn adaptively, such as reinforcement learning.

2.5 Artificial Intelligence Integration

Artificial Intelligence (AI) has kept growing and has been used widely in various aspects such as research, industry, education, and many other areas. The emergence of transformer models in 2017 by Vaswani et al. [34] has boosted the growth of AI, especially in natural language processing (NLP), large language models (LLMs) and the related tasks.

In education, AI has had a huge effect on student's learning achievement in computer science education [31], such as an AI-based instructions to teach online master computer science students [23], AI-based interactive e-book [37], and an adaptable programming tutor [14]. Many other studies also have explored on improving programming education using AI that focus on enhancing the error messages.

Large language models are AI models that have recently become popular due to their powerful performance in NLP tasks. LLMs can be used to generate a more understandable error messages [18], to help students in learning programming. However, several studies that explored on the use of LLMs for generating feedback on programming error messages have reported some negative outcomes, such as vague, misleading and even incorrect results [17, 27, 19]. These poor performance may arise from multiple factors, including model quality, prompt design, and different individual preferences and needs (prior knowledge, learning style, etc.). This could be improved by using the latest and better performing model, better prompt design, and personalization strategy.

For a personalized feedback strategy, an AI model can be applied to effectively recommend the suitable feedback strategy. This can be implemented with reinforcement learning (RL) [28] using the bandit algorithms to adaptively learn and select the effective feedback strategy [5]. Reinforcement learning model learns directly from the action performances, then it uses a "reward and punishment" system to adjust its policy (decision rule). This makes the model always updated automatically and continuously in dynamic environments, thus very suitable as a recommendation system for the personalized feedback strategy selection. This implementation can further improve the personalized feedback system in this study.

3 Methods

This study extends the work of Pomp (2025)[25] on personalized corrective feedback of error message in learning programming, by implementing AI for both selecting the feedback strategy using a reinforcement learning model and generating the enhanced error message using a large language model, and experimenting with more participant number and variety.

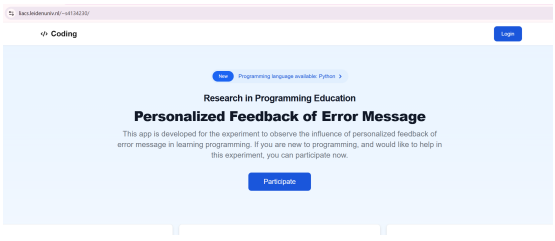

Figure 2: Application user interface.

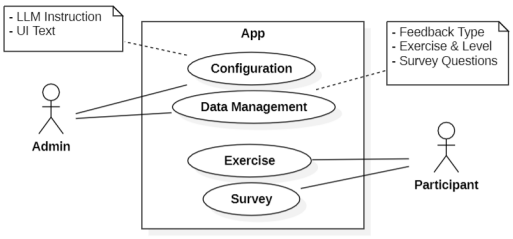

Figure 3: Use Case Diagram.

In this research, an application is developed to experiment with the AI-based personalized feedback system, by simulating a coding exercise, and followed by a survey with Technology Acceptance Model (TAM) [10, 11]. Figure 2 shows the frontpage user interface of the application where its features and functionalities are illustrated in the use case diagram in figure 3.

An empirical study is performed using the collected data from the application logs and the survey responses, to observe the use of the personalized feedback of error message. The target participants are students from higher education who have programming related course in their study, and have little or no computer science background or programming experience.

3.1 Application

The application is a web-based program that simulates a coding exercise and a survey. In the exercise, participants will be asked to solve programming codes that contain error(s) with the help of the personalized feedback of error message, as shown in figure 4. The personalized feedback of error message is an enhanced error message using specific feedback strategy. The system will adjust the personalized feedback strategy based on the participant's performance on the submitted answer. After completing the exercise, the participants will be required to answer a set of survey questions for the data collection of their experiences on the personalized feedback.

A tutorial is provided in the application to guide the user on how to use the application. Participants can access the application through their web browser at anytime and from anywhere.

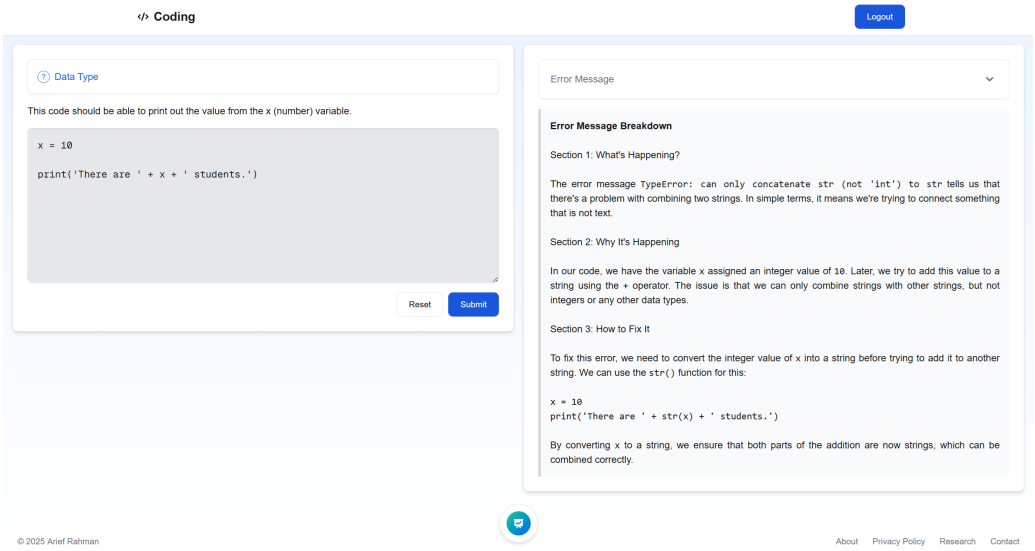

Figure 4: The application explaining the error message.

There are 2 roles and 4 main tasks (cases) in the application as illustrated in figure 3. The admin role is used to manage the list of the exercises, survey questions, feedback strategies, and other configurations. The participant role is the main role that conducts the exercise and survey. The application data are stored in a relational database system using SQLite. It is designed to be flexible so that the contents can be configured and adjusted for further experiment and research.

The application is developed in Javascript using React library with Nextjs framework for the interface (frontend), and using PHP with Laravel framework for the database connection (backend). The source code for the application is available in the GitHub repository (`https://github.com/arief-office/lu_research_frontend`, `https://github.com/arief-office/lu_research_backend`), and is open for further development.

The AI model for the feedback strategy selection is a reinforcement learning (RL) model that uses Gradient Bandit method [28], implemented using Javascript in the frontend side. While AI model for the text generation is using large language model (LLM) from Ollama API service in another server that supports more GPU power. Further details about the AI will be explained in section 3.1.2.

3.1.1 Feedback Strategies

The personalized feedback strategy selection is based on the corrective feedback strategy categories as mentioned earlier in section 2.4 table 1. One of the feedback strategies will be selected randomly, based on the probability distribution calculated by the AI model. The list of feedback strategies along side its probability value are displayed in the application as shown in figure 5. The selected feedback strategy will then be applied to generate the error message and provided to the participant to help solving the error. If the participant fails, the AI will recommend another feedback strategy for generating the error message again. This flow is demonstrated in figure 6.

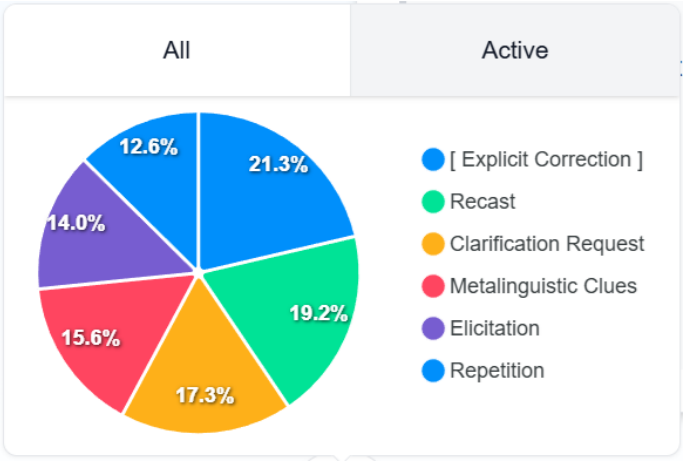

Figure 5: Feedback strategy probabilities from the RL model.

The initial probability distribution of the feedback strategy selection is not equal and adjusted slightly, as mentioned in section 2.4. This is to ensure that the better strategy [20, 13, 24] has a slightly higher chance to be selected, thus could enhance the recommendation process of the personalized feedback. This initial probability distribution can also be changed and adjusted in the application configuration.

To prevent from randomly selecting the same feedback strategy that has been selected previously, that previously selected feedback strategy will be excluded from the next selection, until there are no more available selections. This will make sure that the system will always recommend the new/different feedback if the previous fails.

3.1.2 Artificial Intelligence

There are two AI models implemented in the application. The first is a reinforcement learning (RL) model, to recommend/predict a corrective feedback strategy from the list of feedback strategies. This RL model uses the Gradient Bandit method [28], where in every round/step, the selected feedback strategy will be rewarded (gain points) if the participant succeeds in fixing the code, and punished (lose points) if the participant fails. The feedback strategy with higher points has higher probability to be selected.

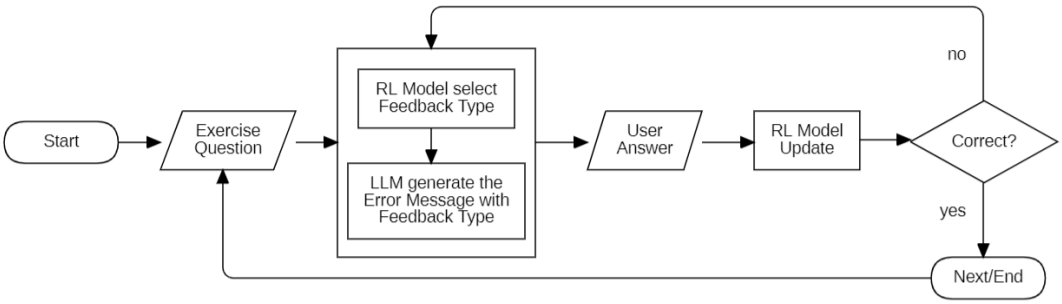

Figure 6: Exercise model flow.

The other AI model is the Large language models (LLMs), that will be used for generating the enhanced error messages with the specified feedback strategy. The LLM will also be used to check whether the submitted answer is correct or wrong.

Reinforcement Learning The list of selections (feedback strategies) is considered as actions, and the RL model (agent) will select one action (a) on every step t. Each action has a preference value $H(a)$, this can be any real numbers, and the initial value is defined from the initial probability distributions. The preference values will then be used to define the new probability distributions P for every step, using the softmax function.

$$P(a) = \frac{e^{H(a)}}{\sum_b e^{H(b)}}$$

The Softmax function normalizes the preference values into probability values of each action $P(a)$ between $0 - 1$, and their sum value is 1 (100% total probability). This function also acts as the policy for the agent. In reinforcement learning, a policy is the rule that will determine how the agent will decide which action to take [28].

On each step t, the action will get R, either a reward or a punishment (negative reward) for selecting an action (rewarding is based on the successfulless of the error message). The reward value R_t will be used to update the preference value of each action $H_{t+1}(a)$ scaled with the learning rate α.

$$H_{t+1}(a) = H_t(a) + \alpha \cdot R_t \cdot \Big[\mathbf{1}(a = A_t) - P_t(a)\Big].$$

- $\mathbf{1}(a = A_t)$ is 1 if a is the selected action, else it is 0

After being updated, the new preference values are used to define the new probability distributions using the softmax function again for the next step. An additional logic is added to improve the model's performance (as explained in the previous section), if the selected action gets punishment, it will be excluded from the list of action selection until there are no more available selections.

Learning rate α is one of the hyperparameter that can be adjusted to improve the model's performance. This and the other parameters can be configured in the application configuration.

Large Language Model The LLM is used in this application for generating the enhanced error message. It is configured with an instruction to explain error messages with the specified rules and feedback strategy. This instruction can be adjusted in the application configuration.

*You are an expert programming tutor. Provide a short and clear feedback to explain the error message in the code to a beginner programmer using feedback type (**selected feedback strategy**)*

Table 2: Instruction prompt for LLM.

Table 2 shows the instruction rules for the LLM. The rules for the feedback strategy use the definition as in Table 1.

Following the instruction, the prompt contains the code snippet that has error, and the raw/native error message. The LLM will then generate the new (enhanced) error message using the defined instruction to be presented in the exercise.

Apart from generating feedback, the LLM is also used for evaluating the submitted answer from the participant. This is very important because the result will be used to adjust the personalized feedback. The instruction for this task can also be configured in the application.

This experiment uses the Meta's pre-trained Llama3.1 model, which is a multipurpose model and capable of understanding code. The model runs on local server using Ollama, a service that can provide LLM API.

3.2 Experiment

The experiment consists of 10 coding exercises with each contains a piece of code that has error(s) related to several concepts in programming, such as Basic Syntax, Variables, Data Types, Operators, Conditions, Iteration, Functions, and other concepts. The exercise can be configured in the application.

The participant can pause and logout during the exercise, and then login again to continue the progress. After completing the exercise, a survey will be conducted.

The experiment takes around 20-40 minutes to complete, with 15-30 minutes for the exercise and 5-10 minutes for the survey.

3.2.1 Participant

The participants consist of 24 students from various study programme. They are students from bachelor and master programmes who have a programming related course and still new to learning programming. A few of the participants don't have a computer science background, and half from the total participants do not have programming experience.

Before participation, all participants are informed about the purpose of the study and provided informed consent. They are also guided to use the application with the tutorial in the application.

3.2.2 Data Collection

The application collects experiment data for the empirical study. The experiment is conducted anonymously, no personal information is collected. The study programme information collected from the participant will only be used for group-level analysis and will not be used to identify an individual.

The application will also collect the student's exercise submissions and the RL model parameter in addition to the survey responses for the data analysis. The collected data is stored in a structured database system.

Technology Acceptance Model The survey uses the Technology Acceptance Model (TAM) [10] [11], which categorizes the questions into 3 components: Perceived Usefulness, Perceived Ease of Use, and Behavioral Intention. This survey model is very effective to analyse the user perception. Each question is measured with a value scaled between 1-5. Table 3 4 5 display the list of survey questions based on TAM.

Code	Item
PU1	The feedback help you understand the error.
PU2	The feedback help you fix the error faster.
PU3	This method help you in learning programming.

Table 3: Perceived Usefulness.

Code	Item
PEU1	This method is easy to use (user-friendly) without needing much technical support.
PEU2	The error message feedback looks clear and concise (not complicated).
PEU3	The error message feedback is easy to follow and understand.

Table 4: Perceived Ease of Use.

Code	Item
BI1	Would you like to use this method of detailed feedback to help you in learning programming?
BI2	Would you recommend your friend to use the personalized feedback?
BI3	Would you like to use the detailed feedback again in the future?

Table 5: Behavioral Intention.

3.2.3 Data Analysis

The application processes the collected data and displays it in the application (accessed using admin role) in a table view, and can be exported to excel or csv format for analysis. The data can also be extracted directly from the database system.

Before analysis, the data is cleaned to ensure completeness and consistency. Incomplete responses and invalid entries will be removed. The data is processed to normalize the scale value where necessary.

The first analysis is by calculating the descriptive statistics to provide a general overview of the dataset. This is done on both the application record/log data and the survey response data. The application log data includes the participant's submission performance, feedback strategy selection frequencies, and the reinforcement learning model parameter. The survey responses contain the participant's background/experience, programme study, and the TAM survey items. This analysis will help for further detailed analysis.

For the TAM survey items, the mean, standard deviation, and frequencies are computed to summarize participants' perceptions of perceived usefulness, perceived ease of use, and behavioral intention. This analysis will help to address the research question "how the students experience and perceive the AI-based personalized feedback in learning programming".

Furthermore, the TAM survey responses will also be used to analyse the implementation of the artificial intelligence for the personalized feedback system in addition to the application log data.

For the reinforcement learning performance, the probability value updates, selection frequency of each feedback strategy, and the convergence will be analysed along side the participant's perception. On the other hand, the performance of the LLM is analysed through the generated error message, participant submission performance, and manual evaluation.

4 Results

A total of 24 participants have carried out the experiment, where the 14 of them do not have computer science background, and 19 of them do not have programming experience. Table 6 displays the participant's exercise submission performances. From the total of 298 submissions, there are 58 incorrect answers. Despite the considerable number of incorrect answers, the number of correct answers 240 indicates that the AI-based personalized feedback system was able to help all 24 students to finish the 10 exercises.

	True	**False**	
Explicit correction	63	6	**69**
Recast	47	9	**56**
Clarification request	28	17	**45**
Metalinguistic clues	26	14	**40**
Elicitation	39	7	**46**
Repetition	37	5	**42**
	240	**58**	**298**

Table 6: The participant's performance on the feedback strategies.

The numbers in table 6 also indicate the occurrence of the feedback strategy that is selected by the reinforcement learning model from its probability distribution. It shows that the most recommended feedback strategy is Explicit Correction with a total of 69 occurrences. This feedback strategy is also considered the most effective as it has the highest number of correct submissions.

4.1 TAM Survey Items

In general, students perceive the personalized feedback positively, as shown in figure 7, the survey results achieved a mean score above the midpoint on all of the items and components of Technology Acceptance Model. However, there is also a relatively high standard deviation, which indicates a slight variation in the score distribution. Although the responses are mostly positive, there are a considerable number of participants that give neutral responses, and some participants perceived negatively.

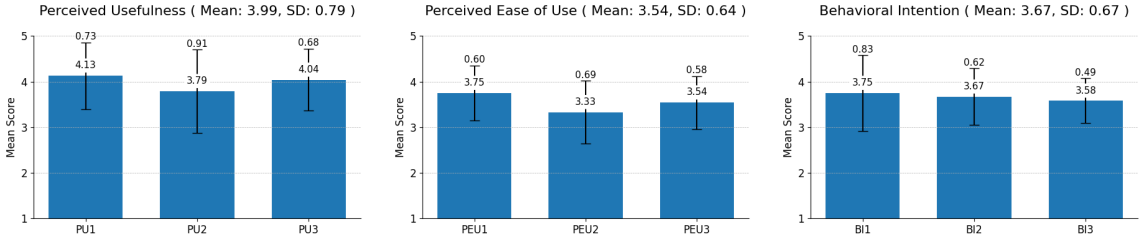

Figure 7: The mean and standard deviation of each TAM survey items.

4.1.1 Perceived Usefulness

The Perceived Usefulness achieved 3.99 mean score, which is a considerably high mean score. This indicates that most participants perceive the personalized feedback to be useful. On the other hand, it also has a slightly high standard deviation (0.79). As shown in table 8, this variation comes from a wide distribution between the different scores, reflecting different "value" of usefulness. A moderate number of participants perceive the personalized feedback as "very useful". However, there are also a few participants (3 out of 24) that do not agree, and perceived this as "not useful".

Score	PU1	PU1	PU3	Total
5	**8**	**5**	**6**	**19**
4	**11**	**12**	**13**	**36**
3	**5**	**4**	**5**	**14**
2	**0**	**3**	**0**	**3**
1	**0**	**0**	**0**	**0**

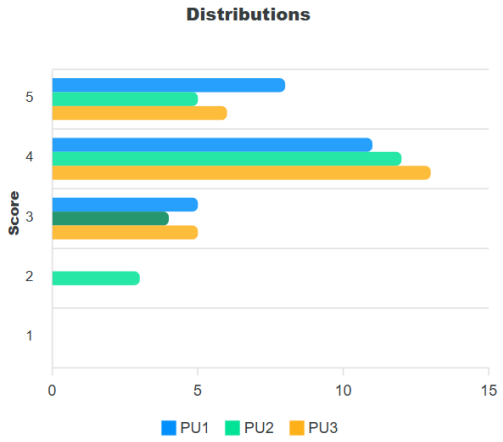

Figure 8: The score distribution of Perceived Usefulness items.

4.1.2 Perceived Easy of Use

The Perceived Easy of Use has 3.54 mean score, slightly above the midpoint. The table 9 shows that there is almost similar score distribution between the positive and neutral responses. Furthermore, there is a very few participant that perceived this system as very easy to use, and there is the same few number of participants that perceive this system as not easy to use.

Score	PEU1	PEU1	PEU3	Total
5	**2**	**0**	**1**	**3**
4	**14**	**11**	**11**	**36**
3	**8**	**10**	**12**	**30**
2	**0**	**3**	**0**	**3**
1	**0**	**0**	**0**	**0**

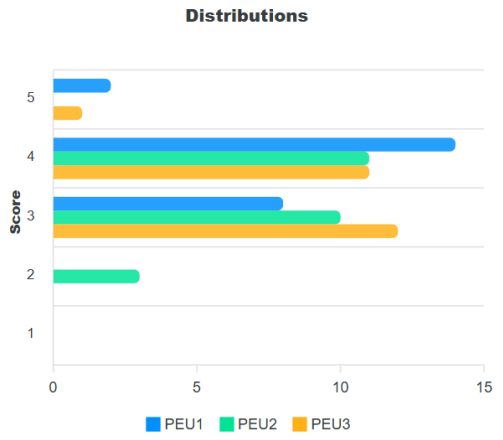

Figure 9: The score distribution of Perceived Easy of Use items.

4.1.3 Behavioral Intention

The Behavioral Intention obtained 3.67 mean score, and 0.67 standard deviation. The score distributions in table 10 shows that beside the higher number of positive responses, there is also a slight number of participants (2 out of 24) that would not like to continue using the system.

Score	BI1	BI1	BI3	Total
5	**4**	**2**	**0**	**6**
4	**12**	**12**	**14**	**38**
3	**6**	**10**	**10**	**26**
2	**2**	**0**	**0**	**2**
1	**0**	**0**	**0**	**0**

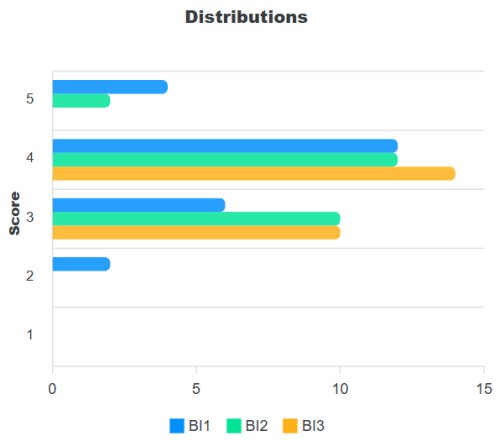

Figure 10: The score distribution of Behavioral Intention items.

4.2 AI Implementation

The reinforcement learning model implementation in the personalized feedback system works relatively well for selecting the feedback strategy. This is indicated by the high ratio in correct answer after failure compared to failure after failure (42:16). However, this performance is also slightly affected by the initial probability distribution of the feedback strategies as listed in table 1.

On the other hand, the large language model performs well for generating the enhanced error messages, despite a very slightly longer processing time around 2-4 seconds. However, it does not perform very well for assessing the submitted answer. It approximately only assess 65% correctly, and was improved to 90% accuracy by designing better prompt.

5 Discussion

This study extends the work of Pomp (2025) [25] on analyzing the influence of personalized corrective feedback of error message in learning programming, by addressing some of the limitations and implementing the artificial intelligence. These limitations include the technical issue in the system during the experimentation, the number of participant, and the variety of the participants. The AI implemented on this research is a reinforcement learning model for the feedback strategy recommendation, and a large language model for the enhanced error message generation.

5.1 Student's Perception and Experience

The results from the TAM surveys show that most students perceive and experience personalized feedback positively. However, there are also a number of participant that do not agree, and there are several factors that could influence their perception and behavioural intention.

The high mean score for Perceived Usefulness shows that the personalized feedback was perceived as helpful in supporting to solve the exercise and learning programming. It suggests that the

generated feedback with the selected strategy was generally effective for most of the participants. This aligns with the previous research by Pomp (2025) [25] that the students perceived the personalized feedback as helpful or useful. However, there is a slight variation in participant's perception. The score distribution in figure 8 shows 3 "Disagree" responses on the item "The feedback help you fix the error faster.". A possible explanation is that the 3 participants have both computer science background and programming experience, hence able to answer the exercise easily, and therefore perceived the feedback as time-consuming.

Perceived Easy of Use has a slightly above midpoint mean score implies that a moderate number of participants perceived this method as complex. This is probably because many of the participants do not have background in computer science and do not have programming experience. Another possible cause is that the generated feedback from the LLM is still difficult to understand for beginner. This is also reflected by the negative perception on item "The error message feedback looks clear and concise (not complicated).".

The moderate mean score for Behavioral Intention indicates that beside the general positive responses, there are also a number of participants that expressed lower willingness to continue using the system. This variation appears to be related to the previous perception. Participants who did not perceive this useful or did not perceive this easy to use are less likely to continue using it.

5.2 Feedback Strategies

The feedback strategy "Explicit Correction" was the most frequently selected strategy, this is probably not only because of the high initial probability, but also due to its "explicit" strategy. This strategy directly provides the correct answer in the feedback. As a result, participants were able to answer the exercise correctly after receiving feedback with this strategy.

In contrast, the feedback strategy "Metalinguistic Clues" is the least frequently selected. This may be caused its implicit strategy, which makes students struggle to understand the error message and failed to answer correctly. This finding is also reflected on the student's perception that show a few number of disagreement.

5.3 Artificial Intelligence

Overall, the use of AI for the personalized feedback of error message demonstrate that the AI implementation was generally effective and received mainly positive perception. However, there are several problems that occurred in the implementation.

The reinforcement learning (RL) model for personalized feedback of error message had performed effectively, indicated by adaptive and steady correct answer after each failure. However, this recommendation method may still have some classic limitations, such as the cold start problem, and the randomness of exploration, which results in the increased time of convergence.

On the other hand, the large language model (LLM) has also performed well in generating the enhanced error message. However, a notable critical issue occurred in the process of answer assessment. The not properly designed prompt for checking the submitted answer may occasionally result in a very low accuracy: a correct answer sometimes assessed as incorrect, and vice versa. This behaviour really affects the performance of the personalized feedback system. Therefore a carefully designed prompt is important for the model to be able to do an assessment.

5.4 Limitations

This study has several limitations that can be improved for further research. First, the exercise questions are too small and limited. More varied exercise questions may improve the experiment

outcome. Second, the participant number is relatively small, this can be increased further and more diverse to better reflect the participant's perception.

One of the most important part of this system is to assess the user's understanding. This is done by assessing the user's submitted answer. The explicitly provided answer from the feedback makes the user to be able to answer the exercise easily regardless of their actual understanding. This can be improved by adding different exercise questions for each feedback generation and the actual exercise question, thus the student's understanding can be evaluated correctly.

Furthermore, the LLM used in the experiment is slightly low quality and outdated. This affects the assessment of the user's understanding. A better quality model may improve the generated feedback and perform better assessment.

5.5 Conclusion

This study analyzed the use of AI-based personalized feedback of error messages in learning programming for higher education students, with focus on the student's perception and technical feasibility.

The results showed that most participants perceived the personalized feedback positively, as indicated by the Technology Acceptance Model (TAM) survey analysis. This suggests that AI-based personalized feedback can be considered acceptable and potentially supportive for novice programmers in helping them to learn and understand programming through feedback of error messages. However, the relatively high standard deviations also indicate a notable variation in each individual perception and experience, reflecting a few negative effectiveness.

The implementation of AI demonstrated both the potential and challenges. The reinforcement learning model was able to adaptively personalize the feedback strategy based on the learner's performance, but it also still face issues such as the cold-start problem and exploration randomness. On the other hand, the large language model was also effective in generating enhanced error messages. However, the inaccuracies in answer assessment revealed that the model quality and prompt design are very important for this implementation.

Future research could improve this study by addressing the limitations on the exercise content, participants, and LLM quality, to further explore the influence of personalized feedback in programming education.

References

[1] M Astia. Corrective feedback in english class. *MA thesis. Minnesota: University of Minnesota*, 2018.

[2] Brett Becker. An effective approach to enhancing compiler error messages. 03 2016.

[3] Brett A. Becker, Paul Denny, James Prather, Raymond Pettit, Robert Nix, and Catherine Mooney. Towards assessing the readability of programming error messages. In *Proceedings of the 23rd Australasian Computing Education Conference*, ACE '21, page 181–188, New York, NY, USA, 2021. Association for Computing Machinery.

[4] Přemek Brada, Ana C. R. Paiva, and Svetlana Tikhonenko. Bachelor-level informatics education in europe: Key data & trends, 2018/19–2022/23, September 2025. CC BY-SA 4.0.

[5] William Cai, Josh Grossman, Zhiyuan Jerry Lin, Hao Sheng, Johnny Tian-Zheng Wei, Joseph Jay Williams, and Sharad Goel. Bandit algorithms to personalize educational chatbots. *Machine Learning*, 05 2021.

[6] T. Charles and C. Gwilliam. The effect of automated error message feedback on undergraduate physics students learning python: Reducing anxiety and building confidence. *Journal for STEM Education Research*, 6:326–357, 2023.

[7] European Commission, European Education, and Culture Executive Agency. *Informatics education at school in Europe*. Publications Office of the European Union, 2022.

[8] Steve Cooper, Jeff Forbes, Armando Fox, Susanne Hambrusch, Andrew Ko, and Beth Simon. The importance of computing education research. 04 2016.

[9] Cristina-Maria Dabu. *Computer Science Education and Interdisciplinarity*. 10 2017.

[10] Fred Davis. A technology acceptance model for empirically testing new end-user information systems. 01 1985.

[11] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q.*, 13(3):319–340, September 1989.

[12] Paul Denny, James Prather, Brett Becker, Catherine Mooney, John Homer, Zachary Albrecht, and Garrett Powell. On designing programming error messages for novices: Readability and its constituent factors. pages 1–15, 05 2021.

[13] Rod Ellis, Shawn Loewen , and Rosemary Erlam . Implicit and explict corrective feedback and the acquisition of l2 grammar. *Studies in Second Language Acquisition*, 28:339 – 368, 06 2006.

[14] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L. Thomas van Binsbergen. Ask-elle: an adaptable programming tutor for haskell giving automated feedback. *International Journal of Artificial Intelligence in Education*, 27, 02 2016.

[15] Noman Islam, Ghazala Sheikh, Ridah Fatima, and Farrukh Alvi. A study of difficulties of students in learning programming. *Journal of Education Social Sciences*, 7:38–46, 10 2019.

[16] Billy Javier. Understanding their voices from within: Difficulties and code comprehension of life-long novice programmers. 1:53–76, 08 2021.

[17] Bailey Kimmel, Austin Lee Geisert, Lily Yaro, Brendan Gipson, Ronald Taylor Hotchkiss, Sidney Kwame Osae-Asante, Hunter Vaught, Grant Wininger, and Chase Yamaguchi. Enhancing programming error messages in real time with generative ai. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–7. ACM, May 2024.

[18] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. Using large language models to enhance programming error messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, page 563–569. ACM, March 2023.

[19] Dominic Lohr, Hieke Keuning, and Natalie Kiesler. You're (not) my type – can llms generate feedback of specific types for introductory programming tasks?, 2024.

[20] Roy Lyster and Leila Ranta. Corrective feedback and learner uptake. *Studies in Second Language Acquisition - STUD SECOND LANG ACQUIS*, 19, 03 1997.

[21] Renée Mccauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. Debugging: A review of the literature from an educational perspective. *Computer Science Education*, 18, 06 2008.

[22] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, (63):81–97, 1956.

[23] Chaohua Ou, David Joyner, and Ashok Goel. Designing and developing videos for online learning: A 7-principle model. *Online Learning*, 23, 06 2019.

[24] ILIANA PANOVA and ROY LYSTER. Patterns of corrective feedback and uptake in an adult esl classroom. *TESOL Quarterly*, 36, 12 2002.

[25] Lisa Pomp. The influence of personalised feedback on coding education for children through the use of an educational game. 01 2025.

[26] Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education*, 18:1–24, 10 2017.

[27] Eddie Antonio Santos and Brett A. Becker. Not the silver bullet: Llm-enhanced programming error messages are ineffective in practice. In *Proceedings of the 2024 Conference on United Kingdom amp; Ireland Computing Education Research*, UKICER 2024, page 1–7. ACM, September 2024.

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2014.

[29] John Sweller. Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, 4(4):295–312, 1994.

[30] D. J Tedick. Research on error correction and implications for classroom teaching. *MA thesis. Samarinda: Mulawarman University*, 1998.

[31] Ahmed Tlili. Can artificial intelligence (ai) help in computer science education? a meta-analysis approach. *Revista Española de Pedagogía*, 82:1–22, 09 2024.

[32] Marina Unterweger, Corinna Hormann, Lisa Kuka, and Barbara Sabitzer. *The Importance of Digital and Computer Science Education in Primary Schools: Perspectives from Educators*, volume 2, pages 545–556. 2025.

[33] Roos Van Amerongen. Error message needs of novice programmers in gradual programming languages. 12 2022.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[35] Iris Vessey. Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies*, 23(5):459–494, 1985.

[36] Sabiha Yeni and Anna van der Meulen. Students' behavioral intention to use gradual programming language hedy: A technology acceptance model. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, ITiCSE '22, page 331–336, New York, NY, USA, 2022. Association for Computing Machinery.

[37] Xiangling Zhang, Ahmed Tlili, Keith Shubeck, Xiangen Hu, Ronghuai Huang, and Lixin Zhu. Teachers' adoption of an open and interactive e-book for teaching k-12 students artificial intelligence: a mixed methods inquiry. *Smart Learning Environments*, 8, 12 2021.

[38] Z. Zhou, S. Wang, and Y. Qian. Learning from errors: Exploring the effectiveness of enhanced error messages in learning to program. *Frontiers in Psychology*, 12:768962, Nov 30 2021.