The Effects of Structured LLM-Generated Feedback on Programming Assignment Performance

Tsvetomila Mihaylova[1], Evanfiya Logacheva[1], Arto Hellas[1], Jing Fan[1], Francisco Castro[2], Bita Akram[3], Narges Norouzi[4], Peter Brusilovsky [5], Juho Leinonen[1]
[1]Aalto University, Department of Computer Science, [2]New York University,
[3]North Carolina State University, [4]UC Berkeley, [5]University of Pittsburgh
Correspondence: tsvetomila.mihaylova@aalto.fi

Abstract

When programming students encounter errors in their code, compiler messages or static analysis output often provide limited guidance, particularly for novice programmers. Personalized feedback from instructors can be effective but does not scale well. Recent advances in large language models (LLMs) enable automated feedback generation at scale. This study examines whether LLM-generated feedback with different levels of guidance is associated with differences in students' problem-solving behavior. We analyze effects on time to solution and number of attempts, and examine whether these effects differ by programming experience. We design three feedback types and compare them to a baseline in which students receive only compiler error messages. Results from an online programming course show that LLM-generated feedback is associated with faster time to solution compared to the no-feedback baseline, with less guided feedback showing slightly stronger effects. Overall, the findings suggest that feedback structure plays an important role in how students progress toward correct solutions and motivate further work on adaptive feedback designs and longer-term learning outcomes.

1 Introduction

When students submit incorrect programming solutions, they typically receive feedback in the form of compiler errors or failed test cases (Keuning et al., 2018). Such feedback indicates that a solution is incorrect but often provides limited guidance on how to proceed, particularly for less experienced programmers (Becker, 2016). Human instructors can offer more targeted feedback, but this does not scale to large courses (Messer et al., 2024). Recent work explored the use of large language models (LLMs) to generate programming feedback automatically (Hellas et al., 2023; Mueller et al., 2025; Vassar et al., 2024; Gabbay and Cohen, 2024; Koutcheme et al., 2024, 2025). Prior studies report that LLMs can produce readable explanations and hints, but also note that generic LLM-generated feedback may be overly detailed, provide complete solutions, or contain inaccuracies (Xiao et al., 2024; Leinonen et al., 2023; Brown et al., 2025). Research in computing education further shows that novice and more experienced programmers differ in how they approach programming problems and respond to guidance (Chuang and Chang, 2024; Kalyuga, 2007).

This study examines how the structure of LLM-generated programming feedback is associated with student performance in an online programming course. Rather than focusing on fully personalized or adaptive feedback, we compare feedback formats that differ in the amount of guidance and information they provide, ranging from more detailed explanations to more constrained, diagnostic hints. We evaluate how these different feedback structures relate to students' time to solution and number of attempts, and examine whether their effects vary across students with different programming experience. In addition, we investigate whether the effectiveness of LLM-generated feedback differs across course content that varies in recency relative to the model training data, which may affect feedback quality.

We address the following research questions: **RQ1**: How does the different feedback complexity affect students' performance, in terms of (a) the expected number of attempts and (b) the time required to solve an assignment? **RQ2**: Do the effects of different hint types on attempts and solution time differ between students with different levels of expertise? **RQ3**: How does the recency of the instructional materials relate to students' performance, as measured by the expected number of attempts and time to solution when using generated hints?

The contributions of this work are as follows:

Proceedings of the 21st Workshop on Innovative Use of NLP for Building Educational Applications, pages 598–611

(1) We design programming assignment hints with different level of cognitive load, motivated by previous work of preferences for different proficiency levels. We show with experiments in an online programming course that LLM-generated hints lead to faster time to solution, with simple hints being more efficient. (2) We evaluate the performance of students of different programming proficiency when presented with the different hint types. (3) We show that new content, released after the LLM knowledge cut-off date, affects the quality of the hints negatively.

2 Background

2.1 Programming Feedback

Automated assessment systems usually provide feedback as test-case results, compiler diagnostics, or static-analysis warnings. This feedback is often difficult for novice programmers to interpret, especially when error messages rely on professional terminology or assume prior knowledge (Becker, 2016; Messer et al., 2024). Enhanced compiler error messages can improve readability, but prior work reports mixed effects on learning outcomes and notes challenges in scaling such approaches across courses and languages (Keuning et al., 2018). Human instructors can provide contextualized feedback adapted to a student's understanding, but this does not scale to large courses. Large language models (LLMs) have therefore been explored for automatic feedback generation. Classroom studies report that LLM-generated hints are often perceived as more readable than compiler output and can support student progress across repeated submissions, while also revealing potential risks of over-reliance on generated feedback (Pankiewicz and Baker, 2023; Leinonen et al., 2022). Other evaluations report that LLM-generated feedback may include overly detailed guidance, complete solutions, or hallucinated issues, and that students often use such feedback primarily for immediate help-seeking rather than deeper self-regulatory strategies (Xiao et al., 2024; Viberg et al., 2025). Designing LLM-based feedback around pedagogical principles influences how students engage with generated hints, with planning-oriented support showing stronger associations with performance than other forms of assistance (Phung et al., 2025). Instructor-based evaluations also indicate that the quality and usefulness of LLM-generated programming feedback depend strongly on prompting strategies and task difficulty, with substantial variation observed across zero-shot, one-shot, and few-shot configurations (Barros et al., 2025).

2.2 Novice vs. Advanced Programmers

Programming behavior differs across experience levels. Novice programmers spend more time debugging, make more syntax-related errors, and rely more on trial-and-error approaches. More experienced programmers make fewer errors and perform more targeted code modifications (Chuang and Chang, 2024; Shrestha et al., 2022). These differences are reported when proficiency is defined by prior experience rather than task performance (Chuang and Chang, 2024). The *expertise reversal effect* describes how instructional approaches that benefit learners with low prior knowledge may become less effective as expertise increases (Kalyuga, 2007, 2009). This effect has been discussed in programming education in relation to the usefulness of detailed guidance at different experience levels. More broadly, adapting the amount and form of instructional explanation to user expertise has been shown to improve learning outcomes in adaptive systems, suggesting that differences in feedback presentation across proficiency levels may be beneficial (Boyle and Encarnacion, 1994). Studies of automated program repair report that novice programmers often struggle to interpret repair suggestions, which may differ substantially from their original solutions, while more experienced learners report greater usefulness; students also tend to value error localization over full repair suggestions (Kurniawan et al., 2023). These studies also report that students often value information about the location of an error and that full repair suggestions can be difficult to apply (Kurniawan et al., 2023). Our work examines whether LLM-generated feedback that differs by programming proficiency and explanation depth is associated with differences in student performance.

3 Methodology

3.1 Context

We applied LLM-generated feedback on an online introductory Web Development course in the span of two months. When a student submits incorrect assignment solutions to the system, by default they see compiler error messages. We assigned at random for part of the users, one of the types of feedback messages. There was an indication that

the feedback was AI-generated, and the students had the option to vote for each message as `Good` or `Poor`.

3.2 Prompt Design

We used three types of prompts (Appendix A), adjusted for different proficiency levels. `General Feedback`: This prompt generates up to three issues in the code, each of which contains description, explanation and hints for next steps. `Single-Issue Feedback`: This prompt contains only one error, with its description, explanation and guidance for next steps towards the resolution. This is motivated by prior work which states that novice programmers could be overwhelmed by the feedback complexity (Xiao et al., 2024). `Diagnostic Feedback`: This prompt contains up to three issues with the code, but gives only descriptions, not hints for how to address the errors. This structure is motivated by the expertise reversal effect (Kalyuga, 2009), where more advanced programmers perform better with more limited guidance. We used GPT-4o for generating all feedback messages.

3.3 Data

The data used in this study is from an online introductory Web Development course. We only used submissions of students who allowed using their data for research. We excluded all exercise-user pairs for which at least one submission is longer than 2000 characters – to keep the prompt length shorter, no feedback is displayed for such submissions. We had three types of feedback, and one control group, (`No Feedback`), which did not see any LLM-generated messages. In addition to the LLM-generated feedback, the students also saw the errors logs from the failed automated tests.

We used the data from two parts of the course – *Part 2: Client-Side Development* and *Part 3: Server-Side Development*. Part 2 contains materials about more recent technologies, which were released after the used model cutoff date, and we refer to it as `New Content`. Therefore we also examined the results per course type, to explore whether the recency of the materials affected the generated hints. We refer to Part 3, which contains content that was used in the model training data, and we refer to it as `Old Content`.

We focused our investigation only on submissions for which the students were actively focused and worked on the task (i.e., engaged). Previous work defines that if a student does not do any input on the programming task for more than 10 minutes, they are likely to be disengaged (Shrestha et al., 2022). Combining this insight together with the distribution of the number of exercises that were solved in particular time intervals, we analyzed assignments that were completed by a student for up to 30 minutes, as nearly 95% of all assignments were solved is in this time interval.

Before starting the course, the students were asked the following question about their proficiency level: *Prior programming experience (1=less than 1 year, 2 = 1-2 years, 3 = 2-3 years, 4=3-5 years, 5=6-10 years, 6=more than 10 years)*, which they were not required to answer. Following previous research (Chuang and Chang, 2024), we split the students as `Novice Students` if their experience was below 3 years (answers 1, 2 or 3 from the above question), and as `Advanced Students` if they had over three years of experience (answers 4, 5 or 6 to the above question). When using this separation, there were 111 `Novice Students` and 38 `Advanced Students`.

After filtering out the submissions as described above, we obtained a dataset with 3153 assignment-user pairs, of which 1228 are `No Feedback`, 665 are `General Feedback`, 639 are `Single-Issue Feedback`, 621 are `Diagnostic Feedback`.

3.4 Approach

3.4.1 Modeling approach

To answer the three research questions, we conducted three experiments, each corresponding to one research question. All analyses were performed on *assignment–user pairs* that were eventually solved, considering submission sequences of up to 30 minutes, as motivated above. Each student sees the same feedback type for all their submissions to the same assignment. For each assignment attempted by the student, we assign the feedback type at random. This design choice was made, in order to not put any student in disadvantage of seeing only one feedback type, before we had any information about the quality of each type.

Because each student may submit multiple solutions to the same assignment and may receive different hints for the different assignments, observations cannot be assumed to be independent. We therefore employ mixed-effects models (Lindstrom and Bates, 1988; Brysbaert and Stevens, 2018), which allow us to account for repeated measures at both the student and assignment levels. Across all

experiments, we include fixed effects for hint type and, where relevant, student level or course part, as well as random intercepts for students and assignments. In addition, we include random slopes for hint type at the student level, allowing the effect of hints to vary across students.

For time-based outcomes, we use linear mixed-effects models (LMMs) (Lindstrom and Bates, 1988; Brysbaert and Stevens, 2018) on log-transformed time. For count-based outcomes (number of attempts), we use negative binomial generalized linear mixed-effects models (NB-GLMMs) (Breslow and Clayton, 1993; Hilbe, 2011) with a log link to account for overdispersion. Linear mixed-effects models were fit using `statsmodels` version 0.14 (Seabold and Perktold, 2010). Negative binomial generalized linear mixed-effects models were fit using `Bambi` version 0.17 (Capretto et al., 2022).

3.4.2 Outcome variables

Each assignment–user pair contains the complete submission sequence up to the first correct solution. From this sequence, we derive two performance measures: **`Time to Success`**: the time elapsed (in seconds) between the first submission and the first correct submission. Due to right skew, this variable is log-transformed for modeling. **`Attempts to Success`**: the number of submissions required until the first correct solution. This variable is discrete and overdispersed and is therefore modeled using a negative binomial distribution.

3.4.3 General model structure

Let y_{ij} denote the outcome for student i on assignment j, and let η_{ij} denote the linear predictor:

$$\eta_{ij} = \beta_0 + \boldsymbol{\beta}^\top \mathbf{X}_{ij} + u_{0i} + u_{1i}\,\mathit{HintType}_{ij} + v_{0j}, \tag{1}$$

where $\mathbf{X}_{ij}$ contains the fixed effects relevant to the experiment, u_{0i} is a student-specific random intercept, u_{1i} is a student-specific random slope for hint type, and v_{0j} is an assignment-specific random intercept.

For `Time to Success`, we model:

$$\log(y_{ij}) = \eta_{ij} + \varepsilon_{ij}, \quad \varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2). \tag{2}$$

For `Attempts to Success`, we model:

$$y_{ij} \sim \text{NegBin}(\mu_{ij}, \alpha), \quad \log(\mu_{ij}) = \eta_{ij}, \tag{3}$$

where $\alpha > 0$ is the overdispersion parameter. The experiments below differ only in the fixed effects included in $\mathbf{X}_{ij}$.

3.4.4 Experiment 1: Overall effect of feedback on performance (RQ1)

To address RQ1, we include only hint type as a fixed effect:

$$\log(y_{ij}) = \beta_0 + \beta_1\,\mathit{HintType}_{ij} + u_{0i} + u_{1i}\,\mathit{HintType}_{ij} + v_{0j} + \varepsilon_{ij}. \tag{4}$$

Here, β_0 represents the baseline performance under the no-hint condition, and β_1 captures the average effect of hint type. The same linear predictor is used for the NB-GLMM when modeling `Attempts to Success`.

3.4.5 Experiment 2: Effect of feedback by student expertise (RQ2)

To address RQ2, students are categorized as *novice* or *advanced*, and we include an interaction between hint type and student level:

$$\log(y_{ij}) = \beta_0 + \beta_1\,\mathit{HintType}_{ij} + \beta_2\,\mathit{StudentLevel}_i + \beta_3\,(\mathit{HintType}_{ij} \times \mathit{StudentLevel}_i) + u_{0i} + u_{1i}\,\mathit{HintType}_{ij} + v_{0j} + \varepsilon_{ij}. \tag{5}$$

The interaction term β_3 captures whether the effect of hint type differs between novice and advanced students. An analogous NB-GLMM with the same linear predictor is used for `Attempts to Success`.

3.4.6 Experiment 3: Effect of feedback across course parts (RQ3)

To address RQ3, we include course part (used as a proxy for content recency) and its interaction with hint type:

$$\log(y_{ij}) = \beta_0 + \beta_1\,\mathit{HintType}_{ij} + \beta_2\,\mathit{CoursePart}_{ij} + \beta_3\,(\mathit{HintType}_{ij} \times \mathit{CoursePart}_{ij}) + u_{0i} + u_{1i}\,\mathit{HintType}_{ij} + v_{0j} + \varepsilon_{ij}. \tag{6}$$

The interaction term tests whether the effect of hint type differs between earlier and later course parts. As before, `Attempts to Success` is modeled using an NB-GLMM with the same linear predictor and a negative binomial likelihood.

3.5 Ethics Statement

The work followed the ethical procedures at Aalto University.

4 Results

4.1 Experiment 1: Effect of hint types on the student performance on the assignment.

`Time to Success` Figure 1 summarizes the fixed effects of hint type on log-transformed time to success with the `No Feedback` condition as the reference level[1]. All hint types were associated with a reduction in completion time relative to the `No Feedback` baseline. The largest average reduction was observed for `Single-Issue Feedback` ($\beta = -0.214$), followed by `Diagnostic Feedback` ($\beta = -0.194$) and `General Feedback` ($\beta = -0.122$). These effects are also reflected in the corresponding fixed-effect estimates shown in Figure 1. Beyond these average effects, the mixed-effects model accounted for substantial between-student and between-exercise variability. Random intercepts were included for both students and exercises, capturing baseline differences in performance. For time to success, the estimated variance of the student-level random intercept was $\sigma^2_{\text{student}} = 0.103$, while the exercise-level random intercept variance was larger ($\sigma^2_{\text{exercise}} = 1.025$), reflecting notable differences in task difficulty. In addition to random intercepts, the models allowed the effect of hint type to vary across students via student-specific random slopes. A wide spread of slopes was observed, which indicates considerable heterogeneity in how individual students responded to hints: while the average effect of hints was beneficial, some students benefited substantially more than others, and a smaller subset showed little or no improvement. This variability suggests that the reported fixed effects capture overall trends, but mask meaningful differences in individual responsiveness to hint support.

`Attempts to Success` The results for attempts, modeled using a NB-GLMM are shown in Table 1. The results indicate that all hint types credibly reduced the number of attempts to success relative to the `No Feedback` baseline, with the largest reduction observed for `Diagnostic Feedback`. The

[1]Note: In all tables with results, we mark in bold the results that were statistically significant, i.e. the Confidence Interval (CI) for MixedLM or Highest Density Interval (HDI) for NB-GLMM excludes zero. Detailed numbers for time to success are given in the appendix.

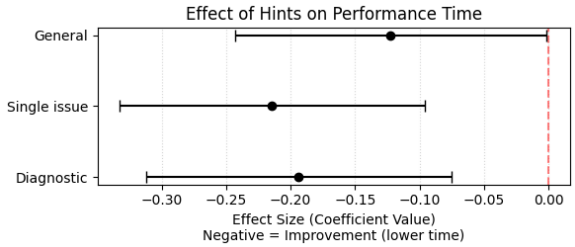

Figure 1: Effects of hint type on `Time to Success`, compared to the `No Feedback` baseline.

Table 1: Fixed effects of hint type on attempts to success (RQ1).

Predictor	β	94% HDI
Intercept	1.216	[1.130, 1.306]
`General Feedback`	**-0.175**	**[-0.238, -0.113]**
`Single-Issue Feedback`	**-0.090**	**[-0.157, -0.030]**
`Diagnostic Feedback`	**-0.200**	**[-0.262, -0.134]**

smallest for improvement is for `Single-Issue Feedback`, which could be explained with the fact that this hint type focuses on one issue only, and therefore might lead to a solution after more iterations. The standard deviation of the exercise-level random intercepts (≈ 0.26) reflects variability in baseline attempts across exercises, indicating differences in inherent task difficulty, while the standard deviation of the user-level random intercepts (≈ 0.17) indicates more moderate variability in baseline attempts across users. The standard deviations of the user-level random slopes for hint type were comparatively small ($\approx 0.05 - 0.07$), suggesting limited variation across users in how the effect of hints on attempts differs.

4.2 Experiment 2: Effect of the feedback for students with different expertise.

`Time to Success` To examine whether the effects of hint type differed between `Novice Students` and `Advanced Students`, we fit mixed-effects models including the interaction between hint type and student level for both `Time to Success`. Figure 2 shows the model-predicted outcomes. `Single-Issue Feedback` and `Diagnostic Feedback` are associated with improvement in the time for completion. In contrast, `General Feedback` does not show a clear effect. The main effect of student level is not statistically significant once hint type is accounted for. Although the interaction terms between hint type and student level are not statistically significant, the predicted trends suggest that the relative effec-

Table 2: Fixed effects of hint type, student level, and their interaction on attempts (RQ2).

Predictor	β	94% HDI
Intercept	1.228	[1.105, 1.334]
General Feedback	**-0.196**	**[-0.325, -0.086]**
Single-Issue Feedback	**-0.220**	**[-0.339, -0.097]**
Diagnostic Feedback	**-0.207**	**[-0.319, -0.089]**
Novice Students	-0.019	[-0.117, 0.078]
General Feedback × Novice Students	0.027	[-0.107, 0.168]
Single-Issue Feedback × Novice Students	**0.174**	**[0.030, 0.316]**
Diagnostic Feedback× Novice Students	0.007	[-0.132, 0.140]

tiveness of hint types might differ between Novice Students and Advanced Students.

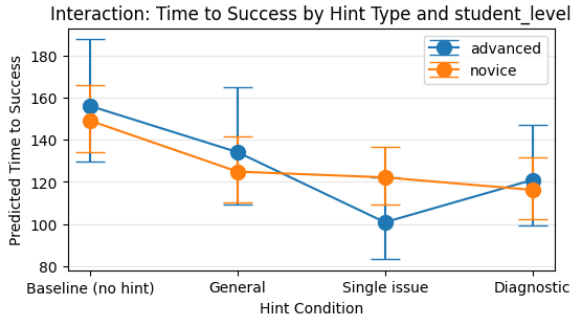

Figure 2: Interaction plot of hint type and student level, with predicted time to success from MixedLM model for assignments solved by up to 30 minutes.

Attempts to Success Table 2 shows the results of the negative binomial mixed-effects model examining the effects of hint type, student level, and their interaction on attempts to success, with No Feedback and Advanced Students as the reference categories. For Advanced Students (the reference group), all hint types were associated with a reduction in the number of attempts to success relative to No Feedback, with posterior mean effects ranging from $\beta = -0.20$ to -0.22 and 94% highest density intervals fully below zero. In contrast, baseline differences between Novice Students and Advanced Students in the No Feedback condition were small and uncertain. From these results it seems that Novice Students do not benefit from a particular hint type, and make slightly more submissions before submitting a successful solution when they see Single-Issue Feedback.

4.3 Experiment 3: Effect of the content recency on the quality of the LLM-generated feedback.

Time to Success To investigate whether the effectiveness of different hint types varies across course progression, we fit mixed-effects model including the interaction between hint type and course part for Time to Success. Model-predicted outcomes are shown in Figure 3. The reference level corresponds to assignments with No Feedback from New Content. All hint types for the Old Content perform better than New Content. All hint types show improvement over the No Feedback baseline for both course parts, with Single-Issue Feedback statistically significant for Old Content. The Diagnostic Feedback performs significantly better for New Content, likely due to the fact that the model could correctly identify errors in the submission, but if it tries to give a direction for fixing, and refers to older version of the material, this could lead to incorrect suggestions.

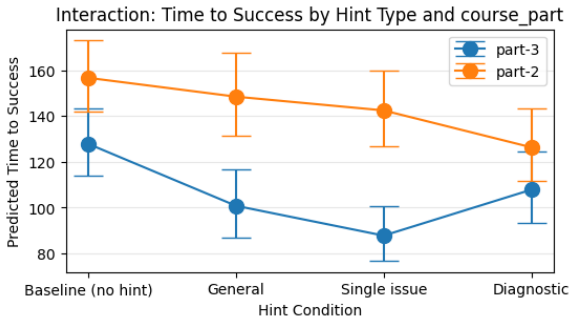

Figure 3: Interaction plot of hint type and course part, with predicted time to success from MixedLM model for assignments solved by up to 30 minutes.

Attempts to Success The results for attempts, modeled using a negative binomial mixed-effects model and shown in Table 3. Unlike the Time to Success, there is no observed significant difference in the Attempts to Success between the two parts when no LLM-generated feedback is shown. For the New Content, again the Diagnostic Feedback is associated with fewer attempts, but interestingly, the General Feedback, which also hints to solution, shows a reduced number of attempts. For the Old Content, the Single-Issue Feedback is associated with fewer Attempts to Success.

Table 3: Fixed effects of hint type, course part, and their interaction on attempts (RQ3).

Predictor	β	94% HDI
Intercept	1.177	[1.060, 1.292]
`General Feedback`	**-0.129**	**[-0.211, -0.055]**
`Single-Issue Feedback`	-0.034	[-0.120, 0.042]
`Diagnostic Feedback`	**-0.217**	**[-0.300, -0.134]**
`Old Content`	0.089	[-0.065, 0.257]
`General Feedback` × `Old Content`	-0.107	[-0.225, 0.007]
`Single-Issue Feedback` × `Old Content`	**-0.131**	**[-0.248, -0.018]**
`Diagnostic Feedback` × `Old Content`	0.037	[-0.088, 0.151]

4.4 Qualitative Analysis

In addition to the quantitative analysis of the student performance in terms of time and attempts to successful submission, we perform qualitative analysis, to identify common errors in the feedback generation. When the feedback was displayed to the students, it was accompanied by a message indicating LLM-generated content, and with option to mark the message as `Good` or `Poor`. Only very few of the feedback messages were evaluated. Most of the feedback that the users voted for was from Course Part 2 (`New Content`). There were only two `Good` and two `Poor` voted exercises for Course Part 3 (`Old Content`). The statistics of the given votes are shown in Table 5. The results in the table exclude one `Good` exercise that was not solved, and one `Good` exercise for which the correct solution was submitted 35 days after the first submission. Non-surprisingly, when `Good` feedback is given, the assignment is solved in a much fewer attempts and for much shorter time than when `Poor` feedback is shown. Students gave the positive vote in the first appearance of the feedback, and they waited for a few submissions before giving negative vote. Table 4 shows how many votes of each kind were given for the different types of feedback, but because of the small numbers, we cannot make any general conclusions from them.

Table 4: Number of `Good` and `Poor` votes by feedback type.

Hint Type	`Good`	`Poor`	**Total**
`General Feedback`	5	8	13
`Single-Issue Feedback`	7	12	19
`Diagnostic Feedback`	2	11	13
Total	14	31	45

We further perform thematic analysis of the `Poor` submission, in order to identify common errors. We first went through all the submissions voted as `Poor` and read them carefully in the context of the assignment description, correct submission and student submission. We also take into account the other submissions by the same user and the same assignment. Then, we coded each voted submission with a description of the type of error it contains. After this, we summarized the codes into common themes. We marked some feedback messages with more than one code, for example in cases where the LLM was referring to an old version, and having had access to the examples in the materials could have showed the expected correct implementation. Several patterns were observed: The suggested fixes were referring to an older version of the taught technology (n=12). The pointed issues could be considered meaningful, but examples in the materials were implementing the functionality in the same way as in the student submission, so there was no actual error that the LLM reported (n=12). The LLM found no issue with the submission, although the tests were failing (n=7). Cases where a wrong issue was proposed, and the real issue in the code was missed (n=4). The feedback was actually correct and the student applied it in their correct submission (n=3).

5 Discussion

Across the course, students receiving LLM-generated feedback reached correct solutions faster than those with compiler output alone, across all feedback types. Differences appear mainly in efficiency (time and attempts), not in final success, as solution rates are high in all conditions. Clear differences emerge between feedback structures. `Single-Issue Feedback` is associated with faster completion but not consistently fewer attempts, suggesting quicker progress without reducing submissions. `Diagnostic Feedback` leads to both fewer attempts and faster completion, indicating more efficient iteration, likely by helping students localize issues. `General Feedback` improves performance over no feedback but shows weaker effects than the more constrained formats. Overall, more detailed guidance does not necessarily improve short-term performance. Limiting feedback to fewer issues or problem identification alone was generally associated with more efficient progress. When accounting for student level,

Table 5: Characteristics of feedback voted as Good or Poor by students. Attempts and time are measured until the first correct submission. Vote position indicates the submission index at which the vote was given.

Vote	#Pairs	#Votes	Attempts to Solve		Time to Solve (s)		Vote Position	
			Mean ± SD	Median	Mean ± SD	Median	Mean ± SD	Median
Good	13	14	2.92 ± 1.04	3	279.10 ± 463.97	155.82	1.65 ± 1.14	1
Poor	22	31	13.95 ± 17.34	7.5	51288.33 ± 87726.14	997.92	19.45 ± 20.71	8

the models do not show strong interaction effects between feedback type and expertise. However, the predicted outcomes suggest that the relative ranking of feedback types differs between novice and advanced students, with feedback appearing more beneficial for Advanced Students than for Novice Students in our experiments. These results indicate potential differences in how students at different levels respond to feedback structure, but they do not provide strong evidence for level-specific optimal feedback. The effectiveness of feedback varied across the different course parts. The results suggest that feedback formats that rely on specific guidance may be more sensitive to limitations in the model coverage of newer technologies. The analyses show substantial variation across students and assignments. Individual students respond differently to the same feedback type, and assignment difficulty contributes strongly to performance differences. As a result, the reported effects represent average trends rather than uniform improvements. This indicates that a single feedback strategy is unlikely to be optimal for all students or tasks. The results motivate further work on adaptive feedback approaches that adjust feedback structure based on student behavior, task characteristics, or prior interaction patterns, rather than relying on fixed feedback formats. Finally, we analyzed LLM-generated feedback evaluated by students and qualitatively examined messages marked as Poor. Due to the small number of votes relative to total submissions, strong conclusions cannot be drawn. Nevertheless, observed failure cases indicate that hallucinations can reduce feedback quality. Providing relevant learning materials in the prompt may mitigate some issues by adding context, especially when the model's knowledge is outdated. Including compiler error messages could also help guide the LLM toward key issues in the submission.

6 Conclusion and Future Work

This study examines how different structures of LLM-generated programming feedback affect student performance in an online course. All feedback types help students reach correct solutions faster than compiler output alone, with differences appearing in efficiency (time and attempts) rather than final success. Feedback structure influences progress: lower cognitive load (fewer issues or less guidance) leads to more efficient short-term problem solving. However, large individual differences between students outweigh effects of expertise level, suggesting that coarse measures like years of experience may be insufficient. Feedback effectiveness also varies with course content. Guidance-heavy feedback performs less consistently on newer material, likely due to outdated model knowledge, indicating that relying only on pretrained knowledge is insufficient for content-specific feedback.

Future work should examine long-term learning effects and consistent feedback strategies across courses. Given high variability between students, more fine-grained personalization should be explored, including adaptation based on task, history, and behavior. Student proficiency could be assessed more reliably (e.g., via diagnostics), and preferences for feedback detail studied independently of expertise. Integrating retrieval or up-to-date course materials into feedback generation may improve accuracy, especially for new technologies. To mitigate the errors caused by LLM hallucinations, automated fact-checking mechanisms could be deployed to verify the correctness of the LLM-generated feedback before it is displayed to the student. Finally, future studies should evaluate long-term learning outcomes beyond short-term performance.

7 Ethical Statement

Displaying of LLM-generated feedback carries all the risks of using output from generative AI. We have taken several precautions to mitigate these

risks. First, the current study was conducted after an ethical approval by our university. Second, we are displaying notifications to the students that the displayed messages are LLM-generated. Third, in order to not put in disadvantage students who only see a particular feedback type, we show different feedback types per student. For the analysis, we only process data from students who have explicitly agreed their data to be used for research. In this work, we explore performance on the task, which does not necessarily correspond to deep understanding. We are planning a longitudinal study to ensure that the displayed feedback actually improves the long-term learning outcomes for the students.

8 Limitations

This study has several limitations. First, students were exposed to different hint types across assignments, so assignment–user pairs are not independent. This design avoided disadvantaging students before knowing feedback effectiveness, but future work could assign a single hint type per student or systematically vary it to study personal preferences. Second, student proficiency is based on self-reported experience, which may not accurately reflect actual ability. Third, the study is conducted in a single course, limiting generalizability, and excludes submissions over 2000 characters, potentially omitting more complex cases. Finally, we focus on short-term performance rather than long-term learning, which should be addressed in future longitudinal studies.

9 Acknowledgments

This work was supported by Research Council of Finland grants #356114 and #367787.

References

Juliana Barros, Laura O. Moraes, Fernanda Oliveira, and Carla A. D. M. Delgado. 2025. Large language models generating feedback for students of introductory programming courses. In *26th International Conference on Artificial Intelligence in Education, AIED 2025, Part 2*, volume 15877 of *Lecture Notes in Computer Science*, pages 421–433. Springer.

Brett A. Becker. 2016. An effective approach to enhancing compiler error messages. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, page 126–131, New York, NY, USA. Association for Computing Machinery.

Craig Boyle and Antonio O. Encarnacion. 1994. Metadoc: an adaptive hypertext reading system. *User Modeling and User-Adapted Interaction*, 4(1):1–19.

Norman E. Breslow and David G. Clayton. 1993. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88(421):9–25.

Neil CC Brown, Pierre Weill-Tessier, Juho Leinonen, Paul Denny, and Michael Kölling. 2025. Howzat? appealing to expert judgement for evaluating human and ai next-step hints for novice programmers. *ACM Transactions on Computing Education*, 25(3):1–43.

Marc Brysbaert and Michaël Stevens. 2018. Power analysis and effect size in mixed effects models: A tutorial. *Journal of cognition*, 1(1):9.

Tomás Capretto, Camen Piho, Ravin Kumar, Jacob Westfall, Tal Yarkoni, and Osvaldo A Martin. 2022. Bambi: A simple interface for fitting bayesian linear models in python. *Journal of Statistical Software*, 103:1–29.

Yung-Ting Chuang and Hsin-Yu Chang. 2024. Analyzing novice and competent programmers' problem-solving behaviors using an automated evaluation system. *Science of Computer Programming*, 237:103138.

Hagit Gabbay and Anat Cohen. 2024. Combining llm-generated and test-based feedback in a mooc for programming. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, L@S '24, page 177–187, New York, NY, USA. Association for Computing Machinery.

Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the responses of large language models to beginner programmers' help requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1*, pages 93–105.

Joseph M Hilbe. 2011. *Negative binomial regression*. Cambridge University Press.

Slava Kalyuga. 2007. Expertise reversal effect and its implications for learner-tailored instruction. *Educational psychology review*, 19(4):509–539.

Slava Kalyuga. 2009. The expertise reversal effect. In *Managing cognitive load in adaptive multimedia learning*, pages 58–80. IGI Global Scientific Publishing.

Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A systematic literature review of automated feedback generation for programming exercises. *ACM Trans. Comput. Educ.*, 19(1).

Charles Koutcheme, Nicola Dainese, Sami Sarsa, Arto Hellas, Juho Leinonen, Syed Ashraf, and Paul Denny. 2025. Evaluating language models for generating

and judging programming feedback. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 624–630.

Charles Koutcheme, Nicola Dainese, Sami Sarsa, Arto Hellas, Juho Leinonen, and Paul Denny. 2024. Open source language models can provide feedback: Evaluating llms' ability to help students using gpt-4-as-a-judge. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, pages 52–58. Association for Computing Machinery, New York, United States.

Oka Kurniawan, Christopher M Poskitt, Ismam Al Hoque, Norman Tiong Seng Lee, Cyrille Jégourel, and Nachamma Sockalingam. 2023. How helpful do novice programmers find the feedback of an automated repair tool? In *2023 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pages 1–6. IEEE.

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, page 885–891, New York, NY, USA. Association for Computing Machinery.

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

Mary J Lindstrom and Douglas M Bates. 1988. Newton—raphson and em algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*, 83(404):1014–1022.

Marcus Messer, Neil C. C. Brown, Michael Kölling, and Miaojing Shi. 2024. Automated grading and feedback tools for programming education: A systematic review. *ACM Trans. Comput. Educ.*, 24(1).

Moritz Mueller, Corinna List, and Michael Kipp. 2025. The power of context: An llm-based programming tutor with focused and proactive feedback. In *Proceedings of the 6th European Conference on Software Engineering Education*, ECSEE '25, page 1–10, New York, NY, USA. Association for Computing Machinery.

Maciej Pankiewicz and Ryan Baker. 2023. Large language models (gpt) for automating feedback on programming assignments. In *31st International Conference on Computers in Education (ICCE'2023)*, pages 68–77.

Tung Phung, Heeryung Choi, Mengyan Wu, Adish Singla, and Christopher Brooks. 2025. Plan more, debug less: Applying metacognitive theory to ai-assisted programming education. In *26th International Conference on Artificial Intelligence in Education, AIED 2025, Part 1*, volume 15877 of *Lecture Notes in Computer Science*, pages 3–17. Springer.

Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, pages 92–96.

Raj Shrestha, Juho Leinonen, Albina Zavgorodniaia, Arto Hellas, and John Edwards. 2022. Pausing while programming: Insights from keystroke analysis. in 2022 ieee/acm 44th international conference on software engineering: Software engineering education and training (icse-seet).

Alexandra Vassar, Jake Renzella, Emily Ross, and Andrew Taylor. 2024. Fine-tuning large language models for better programming error explanations. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*, Koli Calling '24, New York, NY, USA. Association for Computing Machinery.

Olga Viberg, Jacqueline Wong, Yael Feldman-Maggor, Nora Dunder, and Carrie Demmans Epp. 2025. Chatting with code: Exploring llms as learning partners in programming education. In *26th International Conference on Artificial Intelligence in Education, AIED 2025, Part 6*, volume 15882 of *Lecture Notes in Computer Science*, pages 453–461. Springer. Interesting analysis of students talk with ChatGPT from the prospect of SRL. Too many simple requests for the answer.

Ruiwei Xiao, Xinying Hou, and John Stamper. 2024. Exploring how multiple levels of gpt-generated programming hints support or disappoint novices. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–10.

A Prompts and Examples of Feedback Messages

Figure 4 shows the prompts used for the three feedback types. For choosing these prompts, we experimented with several different variants, and ended up with this formulation, which outputs the most concise feedback messages. Examples for the three types of feedback messages are shown on Figure 5.

B Data Stats

The distribution of the number of exercises that were solved in particular time intervals is shown in Table 6.

The solution rates per hint type are shown in Table 8, and show that the hint does not affect whether the students solve the assignment.

General Feedback:

```
SYSTEM
You are a programming feedback tutor. Identify the main problems in the student's code and provide short, direct fixes for each.
Do not show or quote the sample solution.

RULES
- Refer to the student's identifiers.
- Be concise and factual; no filler or repetition.
- Include up to three important issues.
- For each, give a short Problem / Fix / Why block.
- No code snippets or solution hints.
- Prioritize: 1) Spec/I/O issues, 2) Logic/runtime errors, 3) Edge/complexity issues, 4) Style (only if required).
OUTPUT FORMAT
Issue 1:
Problem: ...
Fix: ...
Why: ...

Issue 2:
Problem: ...
Fix: ...
Why: ...
(Up to 3 issues)
CONTEXT
Assignment:
{assignment_description}
Starter Code: {starter_code}
Sample Solution:
{sample_solution}
Student Code: {student_code}
TASK
Give compact feedback listing up to three issues, each with Problem / Fix / Why.
```

Single-Issue Feedback:

```
SYSTEM
You are a supportive programming tutor. Help a beginner fix one main problem in their code.
Do not show or quote the sample solution.
RULES
- Focus on one clear issue.
- Use the student's own variable and function names.
- Keep it short, kind, and easy to follow.
- No code or sample-solution hints.
OUTPUT FORMAT
Problem: (what's wrong)
Fix: (one simple action)
Why: (what it fixes or changes)
CONTEXT
Assignment:
{assignment_description}
Starter Code: {starter_code}
Sample Solution:
{sample_solution}
Student Code: {student_code}
TASK
Give one short piece of beginner-friendly feedback following the format above.
```

Diagnostic Feedback:

```
SYSTEM
You are a peer-level code reviewer. Identify the most relevant issues in the student's code precisely and concisely.
Do not include or hint at the sample solution.
RULES
- Use the student's identifiers when referencing code.
- Be factual and specific; no filler, no praise, no instructions.
- Describe what is wrong, where, and what impact it has.
- Do not restate the assignment or include any code.
- Prioritize: 1) Spec/I/O issues, 2) Logic/runtime flaws, 3) Edge/complexity issues, 4) Style (only if required).
OUTPUT FORMAT
Issues:
- Problem -> Location -> Impact
- (Up to 3 bullets)
CONTEXT
Assignment:
{assignment_description}
Starter Code: {starter_code}
Sample Solution:
{sample_solution}
Student Code: {student_code}
TASK
Give terse, peer-style feedback listing only the key issues as bullets.
```

Figure 4: The three prompt variants used in the study for the different types of feedback.

The descriptive statistics per feedback type are shown in Table 7. In our current setup, each user sees one unique hint type for each assignment that they attempt, but they can see different hint type for the different assignments. This table also shows descriptive statistics about the hint length and complexity, and the weighted Levenshtein distance between the change in two consecutive submissions. The three feedback types differ in their textual properties: `General Feedback` are substantially longer and more variable in length, `Single-Issue Feedback` are shorter and the easiest to read, and `Diagnostic Feedback` are similarly short but more linguistically complex. The average code change between consecutive submissions is similar across hint-based conditions, with `Single-Issue Feedback` resulting in the smallest code changes, which aligns with their focus on addressing one issue at a time. The `No Feedback` baseline shows slightly larger and more variable code changes. Very similar solution rate was observed for all the feedback types, as well as the case where only compiler errors were shown, therefore the different feedback types did not affect whether the students solve the assignment.

```
General Feedback:

Issue 1:
Problem: The `<h1>` element's
↪ text is `"Text
↪ transformations"` instead of
↪ `"Reactive objects"`.
Fix: Change the heading text to
↪ `"Reactive objects"`.
Why: The assignment requires an
↪ exact main heading.

Issue 2:
Problem: The imported component
↪ name
↪ (`Transformations.svelte`)
↪ does not match the
↪ implemented file name
↪ (`ReactiveCounter.svelte`).
Fix: Rename the component file
↪ or adjust the import so they
↪ match.
Why: Consistent naming is needed
↪ to correctly render the
↪ component.

Issue 3:
Problem: The paragraph for total
↪ numbers incorrectly
↪ references `numbers.length`
↪ instead of
↪ `statsState.numbers.length`.
Fix: Use
↪ `statsState.numbers.length`
↪ in the paragraph.
Why: It ensures the paragraph
↪ displays the correct updated
↪ value from the reactive
↪ object.
```

```
Single-Issue Feedback:

Problem: The `if` condition
↪ compares `request.method` to
↪ `PEEK` without quotes.

Fix: Replace `PEEK` with
↪ `"PEEK"` in the condition.

Why: This ensures that the code
↪ correctly compares the
↪ request method to the string
↪ `"PEEK"`, which is necessary
↪ for the condition to work as
↪ intended.
```

```
Diagnostic Feedback:

Issues:

- Incorrect internal link text
↪ -> `<a
↪ href="/contact">Contact</a>`
↪ in line 16 -> The text
↪ should be "Contact page",
↪ affecting compliance with
↪ assignment specifications.

- Unnecessary list wrapping
↪ links -> `<ul>` element
↪ wrapping the `<a>` tags in
↪ lines 15-17 -> Deviates from
↪ instructions, which require
↪ the links to appear directly
↪ below the paragraph without
↪ being in a list.
```

Figure 5: Examples for each of the three variants of feedback used in the study.

C Additional Results

C.1 Experiment 1 Additional Results: Effect of hint types on the student performance on the assignment.

Table 9 shows the fixed effects of hint type on log-transformed time to success with the `No Feedback` condition as the reference level. Figure 6 shows the distribution of the random slope estimates per student for `Time to Success`.

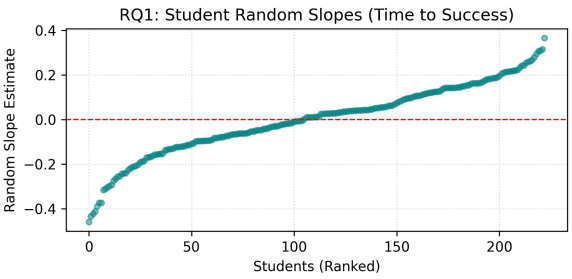

Figure 6: Student random slopes for `Time to Success`.

C.2 Experiment 2 Additional Results: Effect of the feedback for students with different expertise.

The fixed-effect estimates for log time to success with hint type × student level interaction are reported in Table 10.

C.3 Experiment 3 Additional Results: Effect of the content recency on the quality of the LLM-generated feedback.

The fixed-effect estimates for log time to success with hint type × course part interaction are shown in Table 11.

Table 6: Number of assignments that students completed in different time intervals.

Interval	No Feedback	General Feedback	Single-Issue Feedback	Diagnostic Feedback
$\leq$ 10min	1068	587	585	563
10-20min	128	57	41	51
20-30min	32	21	13	7
30min-2h	47	20	15	19
2-24h	16	9	12	10
24-72h	9	3	2	4
$>$ 72h	3	2	6	4

Table 7: Statistics of the data used in the experiments (assignments that were solved up to 30 minutes). Number of assignment-user pairs, submissions, users, and assignments, together with descriptive properties of the generated hints and subsequent code changes by hint type. Values are reported as mean $\pm$ standard deviation. Hint length is measured in words; hint readability is quantified using Flesch Reading Ease (higher values indicate easier readability); code diff is measured as weighted Levenshtein distance between consecutive submissions. There are 47 assignments in total, and each of the hint types was seen for all assignments.

Hint type	**Pairs**	**Submissions**	**Users**	**Hint Length**	**Hint FRE**	**Code diff**
No Feedback	1228	4677	206	–	–	0.074 ± 0.186
General Feedback	665	2208	161	88.78 ± 34.96	54.24 ± 11.12	0.071 ± 0.129
Single-Issue Feedback	639	2312	162	54.23 ± 10.95	57.43 ± 11.22	0.061 ± 0.129
Diagnostic Feedback	621	1966	162	56.30 ± 21.57	44.82 ± 14.42	0.072 ± 0.130
Total	3153	11163	223	–	–	–

Table 8: Number of assignment–user pairs per feedback type, showing total submissions, solved and unsolved cases, and the percentage of solved submissions.

Feedback type	**All**	**Solved**	**Unsolved**	**% Solved**
No Feedback	1328	1303	25	98.1
General Feedback	708	699	9	98.7
Single-Issue Feedback	684	674	10	98.5
Diagnostic Feedback	663	658	5	99.2
Total	3383	3334	49	98.6

Table 9: Fixed effects for log time to success (RQ1). Reference level: `No Feedback`.

	β	95% CI	p
`General Feedback`	**-0.122**	**[-0.243, -0.002]**	**0.046 ***
`Single-Issue Feedback`	**-0.214**	**[-0.333, -0.096]**	**0.000 *****
`Diagnostic Feedback`	**-0.194**	**[-0.312, -0.075]**	**0.001 ****

Table 10: Fixed effects for log time to success with hint type × student level interaction (RQ2).

	β	95% CI
`General Feedback`	-0.151	[-0.411, 0.108]
`Single-Issue Feedback`	**-0.436**	**[-0.697, -0.176]**
`Diagnostic Feedback`	**-0.255**	**[-0.507, -0.002]**
`Novice Students`	-0.046	[-0.259, 0.167]
`General Feedback× Novice Students`	-0.026	[-0.327, 0.276]
`Single-Issue Feedback× Novice Students`	0.237	[-0.064, 0.538]
`Diagnostic Feedback× Novice Students`	0.006	[-0.290, 0.302]

Table 11: Fixed effects for log time to success with hint type × course part interaction (RQ3).

	β	95% CI
`General Feedback`	-0.054	[-0.205, 0.097]
`Single-Issue Feedback`	-0.095	[-0.245, 0.055]
`Diagnostic Feedback`	**-0.215**	**[-0.368, -0.062]**
`Old Content`	**-0.203**	**[-0.339, -0.067]**
`General Feedback× Old Content`	-0.184	[-0.409, 0.042]
`Single-Issue Feedback× Old Content`	**-0.281**	**[-0.503, -0.059]**
`Diagnostic Feedback× Old Content`	0.045	[-0.181, 0.272]