Study Procedure

[Setup] Tasks Folder + Open Protocol + Disable GitHub Copilot + Open PyCharm

- Prepare the tasks folder and test them.
- Open PyCharm before studying to index the Python environment.
- Disable the auto-completion feature of GitHub Copilot.
- For the Chrome Extension task, load the new unpacked folder before the study, and leave it on the `chrome://extensions` page.
- Print the task descriptions on paper for participants.
- Do not open two PyCharm projects at the same time.

[15 minutes] Consent Form + Pre-Study Questionnaire + Background Introduction + Tool Tutorial + Example Task + Zoom Recording

- Use my phone or iPad with another account to connect to Zoom, since the microphone and camera on the laptop are not working.
- Start sharing the Zoom and recording the screen before the first task.

[3*25 minutes] Instruction Reading + Task Performing (Record Subtask Completion) + Cognitive Workload Questionnaire

- Instruct the participants on how to run the code and review the output.
- After reading the instructions, the specified task time is 20 minutes.
- The Diff should be in the unified viewer and trim whitespace.

[30 minutes] Utility Evaluation Questionnaire + Semi-structured Interview

- Begin the speech transcription for the interview.
- The audio recording mostly depends on Zoom, but I will also use my phone to record another version in case it is lost.

[Packup] Data Cleaning + Video Uploading

- Interaction logs path: C:\Users\Lenovo\AppData\Local\JetBrains

Questionnaires (Send to Zoom Chat)

1. Pre-Study Questionnaire
2. Cognitive Workload
3. Utility Evaluation

Background & Tool Introduction

Welcome to our study! Your task is to use our LLM code modification tool to complete three programming tasks. Our study’s goal is to evaluate how you develop your prompting strategies while completing tasks.

Our tool offers two prompting techniques for using LLMs to modify the code. Let's demonstrate each of them via the example task.

- *First, we will provide you with a brief background of the task. [...] Here is the interface. We have two prompting techniques that you can use freely during the task.*
- *The first technique is Summary-Mediated Prompting.*
  - *We can first use the mouse to select part of the code and then click "Retrieve Summary" to get a summary describing its functionality.*
  - *You can not only read it to understand the code, but also modify it to make the code work in the way you intend. For example, regarding the first subtask, [...]*
  - *Then, you can click "Diff Summary" to compare your modifications with the original summary.*
  - *At last, you can click the "Commit" button to request that the LLMs modify the code according to your new summary.*
  - *After receiving the modified code, the diff view will be opened. Please note that you should click the "check" button to accept and the "return" button to reject.*
- *The second technique is the basic Direct Instruction Prompting.*
  - *Select part of the code and directly write your prompt here in a normal way.*
  - *Also, after clicking the "Commit" button, the diff view will open to show the modified code. Click "check" to...*
- *In the interface, the first three buttons correspond to the Summary-Mediated Prompting, while the last button corresponds to the commitment of the action.*

Now, you can try these two techniques in the following subtasks. Feel free to ask me if you have any questions.

Please use our tool as much as possible instead of manual editing. *Ideally, only indentation issues or the removal of irrelevant code can be manually edited. We want to explore your prompting strategies instead of the debugging process.*

Semi-Structured Interview Script

Prior Experience with LLMs

1. Can you discuss your prior experience using LLMs for code modification?
   - Will you rely more on LLM modifications for LLM-generated code?
2. Which tools have you used (e.g., ChatGPT, Claude, DeepSeek, GitHub Copilot, Cursor)?
3. In what programming scenarios have you used them?
4. What challenges have you encountered (e.g., prompting, validation)?
5. Were there moments when you felt more or less in control of the LLM's output, particularly regarding unintended side effects?

Comparison of Prompting Techniques

6. In which cases of modification would you want to focus on understanding code semantics, and in which cases is it enough if the code runs well without needing further checks?
7. Could you discuss the advantages and disadvantages of Summary-Mediated Prompting and Direct Instruction Prompting?

    - Comprehension of code semantics
    - Validation of suggested modification
8. Discuss different factors that impact your choice of prompting techniques:
    - Code scope (large or small)
    - Modification type (insertion or replacement)
    - Intention clarity
    - Task complexity & familiarity
    - Technique/toolkit familiarity

Ways to Use Summary-Mediated Prompting

9. How did you modify the summary to specify your intentions, e.g., by modifying part of the sentence or inserting new sentences?
10. How do you perceive the quality of the generated summaries? Do they help in comprehending the code, scaffolding modification intents, or improving the sense of control over the LLM's output?
11. What drawbacks do you see in Summary-Mediated Prompting? Were there aspects that felt unintuitive or unhelpful? How could it be improved?

In-Study Observations & Industrial Experience

12. I noticed you … when … Could you elaborate on it?
13. [If applicable] As a professional developer who has worked in the industry, how do you think different prompting techniques would affect your programming experience?

Usability Evaluation Items

The prompting technique assists in comprehending the code functionality.
The prompting technique helps scaffold my modification intentions.
The prompting technique is easy to use for specifying desired changes.
I feel that LLM-generated modifications do not introduce unintended side effects.
I generally had a good sense of control over the LLM when modifying the code.
I am confident that LLMs will generate my desired modifications.
I had a good understanding of why the LLM generated such modifications.
I am satisfied with the overall suggestions from LLMs.