CS7180: Sptp in AI: Vibe Coding
Homework 1

Qingyang Zhang
Student ID: 002591724

Due: Tuesday, Feburary 2

1 Prompts making

The most of content of prompts are in a text file with XML labels. All prompts and all versions of code are in GitHub Repository: `https://github.com/YoungZ2357/CS7180-SptpAI-VibeCoding-HW`

1.1 General Prompt

This prompt is used to set limits:

```
Please use the following rules for further development. If there is anyting
unclear about the process, ask me about it. Do not write code until I tell you
 to do so
```

We use this prompt before we upload prompt document. This is essential as those questions serve as one of the bases for refining prompts.

1.2 Prompt Document(Minimal Prompt Template)

The prompt document usually has following structures:

```
<system>
You are a {role here} with {technical depth expectation for LLM. for example,
    with 10 years of experience}
You are developing {content here}
</system>
```

2 Prompt Iteration

The iteration of prompt file has following steps:

1. Write initial prompt document from scratch.
2. Upload initial prompt to LLM along with query template(in which we will require LLM to ask about implemtation detail)
3. Repeat: Answer questions & add more prompts util:1) We are satisfied about the plan LLM gives us; 2)LLM thinks the plan works and stop giving out questions.

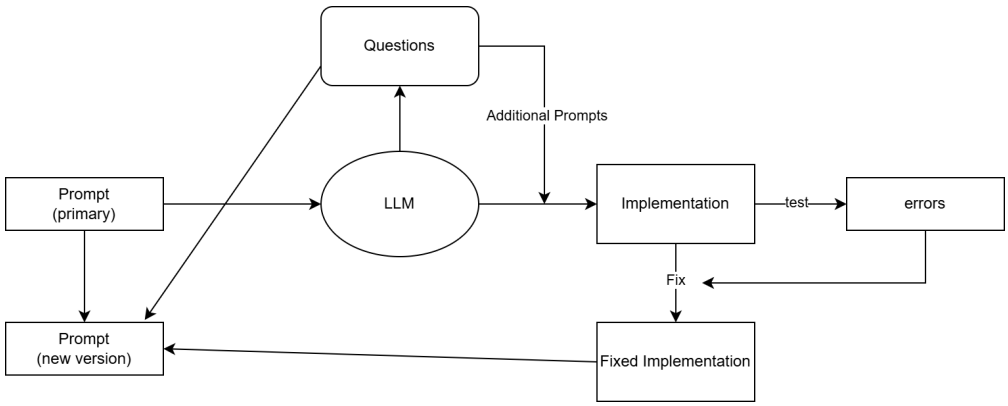

Figure 1: Iteration Process

4. Require LLM proceeds to implementation.
5. Validate code from three aspects:
   - Prompts and initial expectation about this code(for example, modular TypeScript class);
   - Additional expectation after seeing the code.(for example, I want them in separate files/ I noticed that object-oriented is to sophisticated and not needed.)
   - Test result
6. Repeat: Require LLM to create new versions(for example, v1.1 to fix v1.0) of artifact to fix existing problems. If new versions created, require LLM to creat development log based on a give template.
7. When the code is satisfying enough, the conversation stops. Then imporve prompt document and begin a new iteration(Create v2.0 with imporved prompts for v1.0).

3 Code Comparison

Challenge 1: Version1 → Version2

- Prompt change: Added length limits on domain/subdomain. Specified requirements on regular expression. Specified output detail(type).
- Number of additional prompt for older version: around 5
- Code change: Version2 is more modular, separating test script from the main function.

Challenge 2: Version1 → Version2

- Prompt change: Mainly about artifact operations and adding additional functions(manual input for page size)
- Number of additional prompt for older version:
- Code change: The second version features clearer typesetting. Compared to the first version, the manual page size input in the second version is more deeply integrated with the table

display, whereas this functionality in the first version felt more like a patch that was slapped on.

Challenge 3: Version1 → Version2

- Prompt change: Specified file name requirement.
- Number of additional prompt for older version:
- Code change: The second version takes less time to make TypeScript files eligible for compiling. Both versions successfully pass the test using "ts-node" but failed when running on browser.

4 Results

4.1 Tests

Challenge 1

- Version1: All success(Hard encoded edge cases).
- Version2: All success(Hard encoded edge cases). Better code style and reusability.

Challenge 2

- Version1: All components work.
- Version2: All components work.

Challenge 3

- Version1: Test run in console: failed 5 times. Test on browser: Falied.
- Version2: Test run in colsole: Success. Test on browser: Failed.

5 What Makes Good Prompts

Based on observation, most components that make good prompts have already been covered in the lectures, but the specific direction and content require refinement.

5.1 Detailed Description about Output

Possibly, we should always clarify whether we want a piece of code or files. Specifically, we need to give out file names to encourage Claude to use artifacts.**If we don't give out specific file name, Claude is less likely to use artifact, and will give out code in the chatting window** It will also ingore any of your command that requires using an artifact, as it thinks that the content in the chatting window is an artifact.

For example, before Challenge 3 version 1, there is a prompt without output file names and Claude always give out code in the chatting window. After adding file names into the prompt(without changing anyting else), Claude begin to focus on using artifact and asking question on it(artifact versioning, naming and file-artifact matching).

5.2 Repeat Limitations & Ask, Don't Describe

Claude sometimes **ignore requirements mentioned at the beginning of the conversation, even though the conversation is short.** And, if we just give out description(like test result) without a specific requirement, this ignorance is more likely to happen. For example, `"Don't code until I tell you"` is usually ignored.

For example: in Challenge 3, I give out only error messages without questions, Claude immediately created a new version of the artifact, with the error partially fixed. This may stem from Anthropic's emphasis on model diligence, but we should restrict Claude's programming behavior to prevent context hogging and dilution of critical information.

5.3 Describe the Code's Execution/Compilation Process

Languages like TypeScript have different runtime requirements across environments. For instance, while completing Challenge 3, I frequently encountered issues related to ESM, compilation, and localStorage. To address this, we can add elements in prompt document to require "how can we run the code".

```
<requirements>
    <functional>
        <item>
        The code will be tested through "ts-node SmartCache.test.ts"
        </item>
    </functional>
</requirements>
```

A Links

GitHub Repository: `https://github.com/YoungZ2357/CS7180-SptpAI-VibeCoding-HW`

Challenge 1

- Version1: `https://claude.ai/share/7fa640bc-53f0-421a-89e4-d7a1892d0550`
- Version2: `https://claude.ai/share/b478f1e3-ba38-4672-8d69-ed1f7a9da809`

Challenge 2

- Version1: `https://claude.ai/share/9ecef542-1d4c-464b-bfdc-4969fba770cc`
- Version2: `https://claude.ai/share/d58574ea-41ee-45ce-afc2-6d35896cbfc5`

Challenge 3

- Failed Version: `https://claude.ai/share/1272aebe-ed65-443b-b20a-2d55605d7af4`
- Version1: `https://claude.ai/share/d2ead286-71b4-4e73-889b-539148f29650`
- Version2: `https://claude.ai/share/24899aaa-6ff9-41b4-bacc-04eb6ecc7f4f`