Prompt Engineering Documentation (Mind Dash)

Costescu Ioan-Mihail, Rîmboi Eusebiu,
Kononov Daniil, Nițu Teodora

1 Introduction

It is with no doubt that **LLMs** played a major role in developing our project, helping us accomplish a well-structured final product. Tools such as **ChatGPT**, **DeepSeek** and **Gemini** played a huge part in cutting down on the amount of time that we should've actually spent on fixing various subtle bugs, making progress when we were dealing with *programmer's block* or learning how to use a new technology.

First and foremost, it is of paramount importance to understand the context and the full set of circumstances under which we were operating when we started building our project idea. Our team is composed of 4 people - every one of which has numerous qualities and notable talents; however, none of us ever really truly enjoyed having to deal with frontend design, or learning new frameworks on the fly when there are deadlines looming. To further shed light on our arsenal of skills - every single one of us specialized in (or passionate about) data structures, graph algorithms, CodeForces problems and so on.

Thus, we didn't have a lot of experience with *React*, *TypeScript*, or anything to do with front-end design in general. This was our dilemma - we had come up with an idea that showed a really promising future evolution, but simultaneously presented an intimidating obstacle - having to code and study improvisationally, leaving our comfort zones entirely.

2 Using LLMs to study

Some of us (for example, those who really had no prior experience with *React*) made use of **LLMs** by generating study plans, finding useful resources and, overall, basically generating entire *Django* or *PostgreSQL* tutorials on the spot. The main issue that we were dealing with was time - we didn't really have any space to take it slow and spend hours watching YouTube tutorials, so **ChatGPT** helped us familiarize ourselves with a lot of the key principles in these technologies and acted as sort of an assistant.

3 Coming up with ideas and functionalities

3.1 Time Perception Test

LLMs proved to be useful when having to come up with new ideas (for example - designing a test). It goes without saying that we didn't actually copy-and-paste ideas straight from those prompts, and neither did we rely strictly on ChatGPT's ability to simulate creativity.

Howbeit, we did find ourselves in a tough spot after racking our brains to devise a set of about 3 or 4 four tests. At that point, we would essentially ask LLMs what possible skills could such a project attempt to improve (verbal memory, visual pattern recognition, reaction time, etc.), subsequently asking a question the likes of "Give me 10 test ideas for improving one's *skill_name*".

In this manner, we did receive quite a few useful prompts - one of which suggested that we add a **Time Perception Test**. ChatGPT's main idea was the following:

- The test aims to train one's time perception skills.

- When the user starts the test, they have to click on a button when they consider that 10 seconds have passed.
- The true amount of seconds passed is compared with 10 and a result is produced.

We followed this idea with a major change - instead of 10 seconds, we set the time to be a random integer in the interval $[5, 10]$. This improved the test by making it more accurate in its analysis, as opposed to always guessing the same amount of seconds (plus always having to wait for so long).

3.2 Color Memory Test

Another great example would be the **Color Memory Test**. This was (approximately) the main idea of the original prompt:

- Users have to remember a random sequence of colors.
- The colors may repeat (consequently, out of 4 colors, less than 4 may be used).
- The amount of available colors to use for the circles was about 15.

We added the following changes/improvements:

- Instead of using so many colors and shades, we stuck with 7 of the main colors, making the test more enjoyable.
- The colors won't start repeating until the 8th level.

4 Automated testing

Regarding this requirement we had to fulfill, we struggled some time to learn the syntax and understand the concepts and the workflow behind how proficient automated testing is done. But, because of our limited time we allocated on this task, even with efficient task distribution and management, we reached the point where we truly needed help. Unlike the above situations where AI chat-bots proved to be very useful, on this one they failed terribly.

One situation where their limitations were best observable was in mocking, especially fetch-related mocking. Here, the AI tools were wrongly correcting each other such that, at some point, one chatbot suggested me to do some code modification and another one suggested me to either look upon other causes since the problem wasn't there (**there was the problem**) or to undo the work of the previous chat-bot. In this way, we rapidly reached a situation where the AI tools (even the newer, more advanced ones) performed extremely badly.

Although it didn't perform very well, I noticed the new Gemini model provided the most consistent answers and the best advice. With its help, we managed to create the tests for the current project components.