MacGyver Codilla
CS472-1002

Reflection Report on ChatGPT Usage

In the ever-changing world of technology, engineers are compelled to evolve and adapt their toolset for the modern landscape. Generative AI is a relatively new tool for engineers to utilize for tackling complex task. In this lab, I've explored various applications where ChatGPT has been utilized to improve the overall quality of code and documentation. This reflection eport aims to provide insights into my experiences with ChatGPT, diving into the challenges encountered, its impact on my development process, and the lessons I've learned along the way.

Task 1 briefed me with the capabilities that ChatGPT possesses. I initially anticipated that the software development process would be streamlined, however, there were challenges encountered. These challenges were extremely specific to a use-case, and therefore the entire code could not be incorporated into the example engineers code, only a portion of it can be used. Despite this, the development time shortens, and the overall efficiency in the throughput of work is increased. This proves a human must still be present for an engineering task, and ChatGPT is extremely valuable in that human's toolset.

Task 2.1 focused on code refactoring, documentation assistance, and the comprehension of complex pieces of code. In one of my previous projects, I have coded a program that implements a multithreaded approach of the multiplication of two numbers. I have used ChatGPT to examine the code for possible code duplication. As expected, ChatGPT provided suggestions on refactoring the code to alleviate code duplication to aid in readability and maintainability. There were two main functions where calculation took place. One had to deal with the actual multiplication, and the other dealt with the addition step and handling of carry values.

ChatGPT's suggestions were as follows, "The multiplication function performs basic multiplication but also deals with carry values. We can simplify this function to just calculate the result and carry values. We don't need to handle the addition of carry values here; that can be handled separately." Additionally, "Currently, the addition function iterates over each digit position in the partial products, adds them along with the carry value, and handles carryovers. We can refactor this to separate the addition logic from the carry handling."

Repo: https://github.com/39otsu/cs472-group4/blob/main/chatgpt/multithread.cpp ChatGPT Conversation: https://chat.openai.com/share/f0bd9d83-ed7b-4e5d-b984-bc8aa22c9d11

For Task 2.2 I used another of my projects, where I explored the idea of brute forcing and exhausting all combinations of letters to create words that are located in a localized dictionary. This code lacked documentation as it was a quick project in my early computer science education. ChatGPT was utilized to document the code. It provided a succinct description of each function and its purpose.

Repo: https://github.com/39otsu/cs472-group4/blob/main/chatgpt/wordladder.cpp ChatGPT Conversation: https://chat.openai.com/share/8c00b422-463b-4cfb-91ad-db7b45936dbc

In Task 2.3, I then had ChatGPT analyze a previous project of mine that involved graph traversal using Depth First Search to navigate through a map in a videogame. In the map there was a start point, end point, and multiple corridors to traverse through. Since this was a project that was quickly completed, I spent virtually zero time in documenting or commenting it. If one were to revisit this code , the lack of documentation incurs added time in understanding this implementation.

It educated me on best practices for safety, and code convention. Furthermore, it suggested to refactor my depth first search function, and to switch to smart pointers. This refactored code should be more readable, maintainable, and efficient.

Repo: https://github.com/39otsu/cs472-group4/blob/main/chatgpt/dfs.cpp ChatGPT Conversation: https://chat.openai.com/share/c7e12864-bd25-4113-91b3-cc2fbe3848e9

For the last task, I leveraged ChatGPT to improve the CI workflow. The generative algorithm provided insight on areas that have room for improvement. It highlighted the need for code quality analysis, documentation generation, dependency vulnerability scanning, integration tests, and deployment preparation.

Repo: https://github.com/39otsu/tdd ChatGPT Conversation: https://chat.openai.com/share/7826eaf9-0b1d-4e04-a613-f9f8382009bd

Generative AI arms the average engineer with the means to quickly document, analyze, and even generate code in a short amount of time. The advent of publicly accessible generative AI armed the average person gains an additional tool to their arsenal to stay competitive in a highly stressful environment.