Reflection Report

Through the duration of this assignment, the usage of generative AI has proven to be of tremendous aid regarding the creation of both documentation and test cases. After providing very specific instructions on what we were looking for in our software requirements specifications, ChatGPT was able to meet all of those requirements and tailor the documentation to our specific needs. The same applied to the test plans and test cases. ChatGPT even went above and beyond for our test cases; we expected that ChatGPT would simply create test cases, however, it also generated additional documentation such as a test overview, testing objectives, and a test environment. Overall, the generation of the test cases and the software specification requirements thoroughly exceeded our expectations and certainly showcased the strengths of generative AI. However, with strengths come weaknesses.

The most glaring weakness of the generative AI tools that we used was that the code it generated was riddled with syntax errors. Despite the syntax errors, the logic of the code was actually quite sound and easy to follow. This was a surprise to all of us as we had expected to encounter the opposite problem. We were thankfully able to correct the syntax errors without too much trouble because we all understood the logic of the code. In the future, it would likely be more efficient to have ChatGPT generate a skeleton of comments outlining the logic of the code and then manually fill in the rest.

Another weakness that we encountered was trying to generate UML diagrams on ChatGPT specifically. While it is possible to get ChatGPT to generate drawn-out diagrams, as proven by anecdotal evidence of our peers in class, it is incredibly finicky. We attempted to coerce ChatGPT to generate these diagrams for us numerous times to no avail. At first, it gave us some code to use with a tool called PlantUML, which generates diagrams based on this code. While the few diagrams it could generate based on the AI-generated code were great, we encountered issues of the PlantUML code containing syntax errors. Due to our unfamiliarity with this platform, we were unable to diagnose the errors.

Reflection Report

We then prompted ChatGPT to simply draw out the diagrams for us rather than give us PlantUML code. At first, it refused and claimed that it was incapable of such a task. It eventually gave in to our demands and generated a very poorly formatted diagram, complete with blocks of text overlapping each other and arrows pointing in every direction. To remedy this, we ended up using a different LLM model that was able to create accurate UML diagrams based on our SRS that were made using only ASCII characters. This allowed us to easily copy them over and ensured that they were readable.

In summary, our experience with AI for this assignment solidified our understanding of its appropriate uses. Generative AI is a powerful tool for anyone, especially those in a career that involves programming. However, it is just that— a tool. The products of generative AI should be used as a stepping stone to create a final product. While we were able to create a functional program through the heavy use of AI, there was still a large amount of human influence that we had to provide. For example, our SRS was incredibly detailed and tailored to our needs, but that mainly stemmed from giving ChatGPT so much information about what we were specifically looking for. We also had to manually debug the code due to the syntax errors. While we probably could have worked with ChatGPT and had it debug its own code, we felt that it would have been more efficient for us to correct the errors ourselves. Overall, generative AI is a wonderful tool to make programming more efficient, however, it is far from perfect— especially when it comes to generating code— and should be used with caution.