CSC301A1

Jai Dey & Jaivir Singh Panesar

January 30, 2024

Our use of Generative AI:

- FileIO commands in WorkloadGenerator.py, OrderService.java, ProductService.java, and UserService.java were generated using AI.
    - This includes saving and retrieving information from our database. Notably, our program was reading and writing to .json files to imitate the function of an external database.
    - It was also more convenient for us to generate try/catch statements and generate FileReader/FileWriter objects than manually typing this up.
    - Moreover, we used AI to help us with understand how to set the current working directory to the a1 folder. This allowed proper file I/O regardless of which directory each server is started from.
- Our private static Tuple sendHTTPRequest(url, data, command) function was inspired by an AI generated function private static void sendPostRequest(url, data).
    - We modified this function to return a tuple of (string, int) that contained the response and status code from the server.
    - We used if-statements alongside an extra parameter <command> to allow it to send both post and get requests.
    - The AI generated function helped us understand how post requests are sent syntactically in java.
- We used AI to generate a function that converts any input string to its SHA256 hash representation. This was the only function within our project that didn't require any human modifications.

- The layout of the runme.sh file was inspired by AI. We were unsure on how to write switch-case statements in bash scripts, so the syntax was copied over from AI.

- We used AI to help generate the javadoc comments and descriptions. This was arguably the most helpful AI had been in the project. An important takeaway from this is that AI is much better at analyzing and explaining code rather than creating code.

The majority of our code, including the skeletons of all classes, workload parsing, path handling, debugging, error checking, and ideas on how to implement shutdown/restart, was manually written. Of course, we also referred to code provided in lectures as a general guideline for starting the assignment.

Shortcuts:
We decided to cut out the ISCS for A1. This is because our program is relatively small and does not need to handle a large amount of HTTP requests. Instead, we structured our program in a way that the OrderService takes over the functionality of the ISCS. Our OrderService balances and relays HTTP requests to the UserService or ProductService when required. Doing this will require a major rewrite for A2, but our functionality for A1 will not be compromised.

Instead of using dedicated databases such as SQLlite, we decided to store all our User and Product data in .json files. We are already familiar with input and output to files, so it saved us the trouble of downloading, learning, and debugging SQLlite. The downside of this method is that working with files require more "resource expensive" operations than using databases. Fortunately, for A1, we will only be tested with a relatively small amount of data. Using files as our databases will likely be good enough for A1, but it won't scale and will require a rewrite for A2.

Code that will carry over to A2:
Our function private static Tuple sendHTTPRequest(url, data, command) is likely the most useful block of code within our program. It sends an HTTP request to the provided URL and returns the server's response as a tuple. This is a clean and easy way to handle the tricky syntax of java's HttpServer package. Perhaps, more if-statements may be required if we want to expand this function to handle more types of HTTP requests, but doing this only requires a small rewrite.

The process of creating one handler for each type of HTTP request, then branching into ifstatements within each handler's body, and finally returning a response to the client is a very solid way to build a server. It is very likely that we will be using a similar layout for A2 with minimal changes, if any. Likewise, we predict that the skeleton of our UserService and ProductService classes will be very helpful for A2.