CSC301 Assignment 1
Write Up

Daniel Steven
Simon Hunter

Which sections of your codebase used Generative AI?

We used Claude (claude.ai) to generate a lot of the HttpServer and HttpConnection boilerplate code. This was our first time working with these libraries, and it seemed a bit too much code to take initially. However as the assignment progressed, we grew more comfortable with the library and a lot of the end codebase was heavily modified from Claude's generation.

There is unmodified GenAI code in our Shared package, which is a bunch of shared tools used by all our microservices. These are things like our DataBase implementation, PasswordHashing, and other tools. Some of these tools used libraries outside of the scope of the course, which would take too much time to learn (particularly our ScuffedDatabase, SecurityUtils and HttpUtils classes).

We also used GenAI to learn abstract concepts, like the best design patterns, logic and control flow, and widely used best practices for setting up a microservice HTTP backend. As students, we had no idea, conceptually, on what an "optimal" design for a web server would look like, so we asked Claude.

What "shortcuts" were taken during this Assignment?

Our database (class ScuffedDatabase) was a major shortcut taken during this assignment. During the development process, all the microservices were written to run on temporary memory. When the new requirement was revealed one week into the assignment, we didn't want to break the code that had been working already. So, we made a class that just writes the entire temporary memory to file every time the memory changes. This strategy satisfies a small database, but will be too slow for A2, which can have thousands of objects of data.

Another shortcut was the ISCS. It literally just forwards HTTP requests with no other features available. Currently, it assumes that there is a single instance behind each service on the backend, which will change for A2 as the number of users and products scale up, and services must be distributed. This entire service will probably be rewritten in Assignment 2 to implement load balancing and other features.

Which sections of code will not require a major rewrite for Assignment 2? Justify Why.

We believe that UserService and ProductService will *not* require major changes in A2. The reason being that the core functions of these processes (creating an object, updating an object, and deleting an object) are not affected by scalability. Assume that we rewrote these services so that they are distributed. Each distributed process will still have to execute the same business logic - which is carryover from the previous assignment. The API endpoints for these services also won't change. Only minor changes need to be made, like distributing the service, and controlling resource access.

Some of the shared utilities we made like the PasswordHasher and HttpUtils class won't need to be rewritten. This is because they contain linear "atomic-like" actions that can not be sped up by parallelization.