CSC301 ASSIGNMENT 1 WRITE-UP

Generative AI Usage in Our Submission

We utilized Generative AI in several parts of our project to assist in understanding the assignment requirements, generating base code, and refining our final write-up. Specifically:

- **Service Implementations (ProductService, UserService, OrderService)**: We used ChatGPT to generate initial code structures for these service classes. As we began testing and debugging, we iteratively modified the AI-generated code to better fit our project's requirements. We also used Claude to write most of the edge case functions, after having written a few for each service, so that it had a structure to follow for the rest of the functions. The functions were modified to fit assignment specifications.
- **Write-up Refinement**: After drafting this write-up, we used ChatGPT to improve clarity and readability. We then made additional edits to ensure it accurately reflected our thoughts and maintained our writing style.
- **OrderService and ProductService Integration**: To connect OrderService with ProductService, we sought help from both ChatGPT and Claude Sonnet to understand the necessary implementation steps. The initial AI-generated code remained largely unchanged, as modifications often led to crashes. We only made minor adjustments for formatting and readability. GenAI helped us overcome the difficulty between creating a steady data flow between the various services.

Shortcuts Taken in the Implementation

Given time constraints, we had to make certain compromises in our implementation:

- **Skipping ISCS File Implementation**: Due to limited time and the need to complete writeup.pdf, Javadocs, and runme.sh, we opted not to create a separate ISCS file. Instead, we integrated the necessary API handling directly into OrderService, allowing it to call ProductService and UserService as needed. We understand this will impact/worsen the difficulty of implementing a larger scale version of assignment 1 in the future.
- **OrderService Efficiency Trade-offs**: OrderService currently connects to ProductService to validate orders, checking if the product exists and if there is enough stock. While functional, this approach may not scale well for A2 when handling a larger number of users and commands. We anticipate revising this design to improve efficiency, probably handling it in the ISCS file.
- **Handling System Shutdowns**: In the future, we may need to implement functionality to manage the shutdown and restart of individual service instances. Currently, our system does not account for these scenarios, and we expect to revisit this for A2. We intend to implement some sort of persistent storage system to take care of this scenario.

Code Components That Likely Won't Require Major Rewrites

Some parts of our submission are expected to remain mostly unchanged in A2:

- **Parser.py**: This file primarily reads input and passes it to OrderService. Its functionality is straightforward and unlikely to require significant modifications.
- **ProductService and UserService**: The core logic of these services is stable, as their fundamental roles will not change. However, if we implement machine shutdown handling in A2, some minor adjustments may be necessary to ensure system stability and transaction continuity.

Overall, while some components will need to be revisited for scalability and efficiency, the foundational structure of our project is in place. We made strategic decisions to balance functionality with time constraints, ensuring a working submission while identifying areas for improvement in A2, leaving some flexibility in code architecture for the future.