For the UserService and ProductService, we used Generative AI for the database connection code, especially the “initDatabase” function which provides a framework for database schema. The rest of the ProductService code did not use Generative AI. Some short snippets of AI generated code are scattered throughout the user service, particularly the database queries.

Most of the code in the ISCS was written by Generative AI with minor changes. We used a prompt similar to “given our product service endpoints… and our user service endpoints, how to redirect order service requests to these endpoints? Code it in ISCS class”. One of the important changes we have made to the ISCS file is the way we read the request body from upstream services(product, user): we use getInputStream to get response body from 200 status code response and getErrorStream to get response body from failed response.

For the OrderService, much of the code involving HttpURLConnection and HttpExchange are AI-generated. Quite some modification was needed though, since the AI-generated code was buggy and did not handle errors and edge cases well.

For workload_parser.py: We used Generative AI on send_requests function as a quick way to switch the mindset of dealing network requests from Java to Python. This file need to be rewrite in A2 as we only handled a limited set of edge cases in workload inputs, and we hardcoded some placeholder values in case some field is missing(e.g. Any missing integer value is represented as either empty string or -4321). Both empty string and -4321 will trigger the server to return an empty JSON object and correct status code which will likely pass the test cases, but it does not truly reflect defective input though.

We believe both UserService and ProductService codes do not require major rewrites in A2 as they follow the instruction given and perform CRUD operations to the database only. The OrderService probably won’t need any major rewrites either. New API-endpoints and functionality can be integrated without changing much of the current codebase.

The ISCS will need major modification, though. Currently, it only relays HTTP requests from the OrderService to UserService and ProductService without modification. If the system is to scale in A2 efficiently, load-balancing would need to be implemented in the ISCS.

Our current implementation of A1 uses SQLite databases. These might also need to be switched to more scalable databases in the future, for example databases with actual connection(PostgreSQL, etc) instead of a .db file, if we need to scale each service to multiple machines.

Last, we use Generative AI on our custom test script generation to accelerate testing with JSON datasets input given in starter code.