Writeup

Ziao Liu, 1006096797
Chengxi Li, 1008554734

January 30, 2024

Explanation

Using ChatGPT for this assignment significantly enhanced our productivity and learning curve. This project presented a unique challenge due to its flexibility and the minimal starter code provided, placing us in a position where we had to learn everything. Initially, we had a basic understanding of the project's requirements but were unsure about the implementation specifics, especially concerning server-client interactions and HTTP requests. Our familiarity with the required libraries was limited, making the extensive documentation and myriad of methods overwhelming. ChatGPT proved invaluable in this context, offering clear guidance on the appropriate methods to use and their implementation. This assistance was crucial in completing the user service component, after which we confidently adapted the learned patterns to develop the product service independently. ChatGPT also played a major role in debugging. A notable instance was when we encountered issues with a database interaction within a delete request. Despite our efforts to diagnose the problem using print statements, the issue persisted until we consulted ChatGPT. The AI quickly identified the absence of an "AND" statement in our SQL query, leading to an immediate resolution. This experience underscored ChatGPT's role not just as a tool for generating code snippets but as a dynamic resource for problem-solving and learning. It facilitated a deeper understanding of programming concepts and best practices, enabling us to approach similar challenges with increased confidence and efficiency.

Here's an example of how we harnessed ChatGPT's capabilities and tailored the output to our needs:

```
try (Connection conn = DriverManager.getConnection(url);
     PreparedStatement query = conn.prepareStatement("SELECT * ...
         FROM users WHERE id = ?")) {

    query.setInt(1, id);  // Assigning the user ID to the query ...
        parameter
    ResultSet rs = query.executeQuery();  // Executing the query

    if (rs.next()) {
        String name = rs.getString("username");
        String email = rs.getString("email");
        String password = rs.getString("password");

        JSONObject response = new JSONObject();  // Constructing ...
            the response JSON
        response.put("id", id);
        response.put("username", name);
        response.put("email", email);
        response.put("password", hashPassword(password));  // ...
            Hashing the password for security

        sendResponse(exchange, response.toString(), 200);  // ...
            Sending the response back to the client

    } else {
        // Responding with an error message if no user is found
        sendResponse(exchange, "User not found", 404);
    }
}
```

In this segment, ChatGPT initially provided a foundation on how to construct and execute SQL commands within a Java environment. We then refined this foundation, ensuring the code adhered to best practices and the specifics of our API documentation. This process not only facilitated database interactions but also solidified our understanding of SQL execution in Java, proving to be an invaluable learning experience.

While ChatGPT significantly enhanced our efficiency and problem-solving capabilities, it's important to approach such tools with a mindset geared towards learning and understanding. Relying solely on AI-generated solutions without comprehension can hinder one's development as a programmer. Thus, we advocate for using ChatGPT as a catalyst for growth, leveraging its insights to deepen one's knowledge and skillset.

In A1, we took some expedient measures that might not scale well for A2. One significant shortcut was not developing a dedicated inter-service communication (ISC) system. Given A1's scope, where we primarily managed requests, using the order service as a makeshift ISCS seemed sufficient. However, the complexity and volume of A2, which involves handling up to 1 million requests, necessitate a robust ISCS system. The ordered service, under such load, might become a bottleneck, struggling with both its intrinsic functions and mediating between services. To address this, A2 will require a dedicated ISCS service that not only facilitates efficient message passing and load balancing but also supports advanced features like multi-threading for concurrent processing.

Moreover, our initial implementation lacks comprehensive synchronization mechanisms. This oversight could lead to data inconsistencies or system crashes under concurrent operations, a risk that becomes pronounced with the scaling requirements of A2. Implementing proper synchronization and concurrency control is essential to ensure system stability and data integrity.

Our current approach to service lifecycle management, particularly handling "shutdown" and "restart" commands, is also too simple. We rely on a global flag to check for the first command post-restart and decide whether to drop and reinitialize database tables. This approach risks data loss if a service is shut down without a subsequent "restart" command, an issue that becomes critical in A2 where service resilience and data persistence are paramount. Developing a more sophisticated strategy for data preservation and service recovery is crucial to accommodate unexpected shutdowns or failures, ensuring continuous operation and reliability.

For A2, both the user and product services, due to their straightforward responsibilities, might only require minimal adjustments to accommodate new features. However, the overarching system architecture, including ISC, synchronization, and service lifecycle management, will need significant redesign and enhancement to meet the heightened demands of scalability, reliability, and performance.

Lastly, the workload parser should also be refined to support multi-threading, being able to send concurrent requests to the services to reduce the total runtime.