Ticket Selling System

ChatGPT Implementation Report

In this experiment, ChatGPT-4 is used to implement the ticket-selling system that includes the server and client programs. The server program provides ticket services such as buy and sell to two local ticket distributors, which is executed by one client program. When executing the server and client programs, both clients are to send 15 automatic buy requests to the server via a TCP connection. The server is to wait for both clients to connect and respond equitably to their requests. If the client has sufficient funds and there is at least one ticket available for sale, then the server will send the ticket number and its price, and update the ticket database. If the client does not have sufficient funds, the server will send a message indicating so to the client. If the tickets are sold out, the server will send a sold-out message to the client.

As for the clients, if one runs out of money, that client will sell back one ticket to the server to get the funds for completing the 15 automated buy requests. As the ticket is sold to the server, the client needs to update his database and balance accordingly. When a client receives a sold-out message from the server, that client will send a buy request to the scalper, which is the other client. The clients are connected to each other via a UDP connection. The scalper will resell a ticket at double the original price to the other client. If the client has enough money to buy a scalping ticket, then the client and the scalper will update their ticket database and balance accordingly. If the client does not have enough money, then he will resell his ticket to the server at the original price and update his database and balance accordingly. After completing the 15 automated buy requests to the server, the client will print out the current state of his database along with the remaining user balance and close the TCP connection. Before closing the connection and exiting the program, the server must wait for both clients to close the TCP connection, and then the server must print out the final state of the ticket database.

As this system requires multithread programming and process synchronization with different execution logic for both the client and server, ChatGPT was prompted with the description of the server and client separately and was asked to outline the steps required for implementing the functionality of each program. ChatGPT was asked to implement the base of the server program given the first three steps in the outline. As an evaluator, I evaluated the base program and executed it to make sure it had no error. Then, ChatGPT was asked to implement the rest of the outline steps for the server program, which was evaluated and compiled by me. The same procedure was conducted to generate the client program.

Based on the previous experiment approach, asking ChatGPT to implement everything at once can leads to complications such as missing required functionalities, lots of errors in both programs (costing more time for debugging), and unintended behaviors in both programs. Therefore, I decided to evaluate the basic functionalities (buys and sells) of the server and client programs and their interactions before implementing the scalping functionalities on the client program. The prompt to generate the program along with the outputs and comments are detailed in the Experiment Tracking Sheet and stored in my GitHub.

Through 10 iterations, 6 errors occurred while executing the server and client programs:

1) The server did not wait for both clients to connect before selling tickets. → Fixed.

2) Client 1 finished first and exited, the server exited, and Client 2 was still idling indefinitely waiting for the thread handling UDP connection to complete. → Not fixed: the issue is found to be not on the server side but is related to error 4.

3) In a message containing the ticket ID and price, the server sent: "1000 3411000 341", causing an error of too many values to unpack. → This issue only happened while executing server-0-5.py and client-0-6.py.

4) Error in UDP connection: client-0-6.py", line 29, in udp_listener udp_sock.bind((hostname, udp_port)) OSError: [WinError 10048] Only one usage of each socket address (protocol/network address/port) is normally permitted. → Not fixed

5) The server program lost some of the original requirements that were already implemented in server-0-7.py. Ex: The server did not print the initial database. The server did not wait for both clients to connect before selling tickets → Fixed in server-0-7.py but occurred again in server-0-8, and fixed in server-0-9.py.

6) The server program server-0-9.py missed a required functionality, which handles NOFUNDS when the client does not have enough money to buy a ticket. (The NOFUNDS condition was already implemented from server-0-5.py to server-0-7.py). → Fixed in server-0-9.py

The first error was fixed in program server-0-7.py, but it occurred again in server-0-8.py when ChatGPT was asked to address error 2. This loop of errors also happened in the previous experiment approach when ChatGPT was asked to fix an issue, and it caused another issue. This newly occurred issue most likely already happened and had been fixed before that.

To address error 1 again, the server-0-7.py was attached as an input to generate server-0-9.py. However, server-0-9.py missed some of the required functionalities (errors 5 and 6), which were already implemented in server-0-7.py. This leads to the question of whether ChatGPT considers the input file server-0-7.py before generating server-0-9.py, or whether ChatGPT generated the implementation based on the server description at the beginning and forgot some of the requirements.

My next step is to address the remaining errors before implementing the scalping function on the client program. Once both programs are executed as expected, I will analyze the program codes using program analysis tool and generate test cases to verify their correctness.