**CIS427-Programming Assignment 2**

**Online Pokémon Cards Store**

Jacob Marchywka, Meher Jabbar, Andrew Leutzinger, Brandon Bailey

**Project Description**

This project implements a client-server, Online Pokemon Cards Store application. The two main programs, client.py and server.py, utilize network sockets in the Internet domain to communicate with each other. The clients send requests to the server, while the server handles multiple clients by creating threads for each client attempting to access the database.

**Environment**

The code is compatible with the current latest version of Python, 3.12.0. Ensure you are using Python3 on your device. To run the code, an SHH client software will need to be utilized. For Mac and Linux you can run the ssh client in Terminal. For Windows, the Bitvise software may be used.

**Student Roles**

Brandon Bailey – ‘LOOKUP’ as well as contributions to ‘LOGIN’

Meher Jabbar – ‘DEPOSIT’ and ‘WHO’, with contributions to the rest of the code

Jacob Marchywka – Update existing operations from Assignment 1, ‘LOGOUT’, as well as contributions to the entirety of the code itself

Andrew Leutzinger – ‘LOGIN’ and ReadMe document.

**Usage**

The client and server can be implemented directly in the command-line. To do this, type the following commands into 2 separate terminals, starting with the server that will start listening for client connections:

python3 server.py

python3 client.py

Then from the client you must enter the server IP address. Once connected, the client must LOGIN to the server with the correct username and password. From there, you can use the BUY, SELL, LIST, BALANCE, WHO, DEPOSIT, LOOKUP, QUIT, SHUTDOWN, and LOGOUT commands to communicate with the server. Below are details on each of the commands, including formatting, examples of a success client-server interaction, and functionality:

**LOGIN**

LOGIN takes the username and password input form the command line and checks to make sure these match the database. Once access is granted, the client may access all commands besides shutdown. Accessing commands is only possible once the client is logged in. Below is an example, where the format is ‘LOGIN username password’

*Client: LOGIN JohnDoe Password*

*Server: Received: LOGIN JohnDoe Password*

*Client: 200 OK Welcome!*

**BUY**

BUY allows the client to purchase cards existing in the database. Below is an example, where the command format is 'BUY card\_name rarity price number\_of\_cards user\_id':

*client: BUY Pikachu Electric Common 19.99 2 1*

*server: Received: Pikachu Electric Common 19.99 2 1*

*client: 200 OK*

*BOUGHT: New balance: 2 Pikachu. User USD balance $60.02*

**SELL**

SELL allows the client to sell cards that it currently owns. Below is an example where the command format is 'SELL card\_name number\_of\_cards price user\_id':

*client: SELL Pikachu 1 34.99 1*

*server: Received: SELL Pikachu 1 34.99 1*

*client: 200 OK*

*SOLD: New balance: 1 Pikachu. User’s balance USD $95.01*

**LIST**

LIST allows the client to view only its own records. However, if the client is the root user, LIST will show all records for all users. Below is an example where the command format is 'LIST' under user John Doe:

*Client: LIST*

*Server: Received: LIST 4*

*Client: 200 OK*

*The list of records in pokemon cards table for user, John:*

*ID Card Name Type Rarity Count Owner ID*

*6 Pidgey Flying Common 2 4*

*7 Squirtle Water Common 3 4*

**WHO**

WHO allows the root user to see what other users are currently active by displaying their UserID and IP addresses. The root user is the only one who may access this command. Below is an example of a root user’s command being ‘WHO’:

*Client: WHO*

*Server: Received: WHO*

*Client: 200 OK*

*The list of active users:*

*John 141.215.204.179*

*Root 127.0.0.1*

**LOOKUP**

LOOKUP allows the client to search for a card in the database by partial or full card name. The server will then send a list of all matching cards. Below is an example where command format is ‘LOOKUP Fire’:

*client: LOOKUP Fire*

*server: Received: LOOKUP Fire*

*client: 200 OK*

*Found 1 match:*

*ID Card Name Type Rarity Count Owner*

*3 Charizard Fire rare 1 1*

*Client: LOOKUP Random Card*

*Server: Received: LOOKUP Random Card*

*Client: 404 Your search did not match any records*

**DEPOSIT**

DEPOSIT allows a user to add more funds to their records in USD. Below is an example where command format is ‘DEPOSIT 27.00’

*Client: DEPOSIT 27.00*

*Server: Received: DEPOSIT 27.00*

*Client: Deposit successfully. New User Balance $127.00*

**LOGOUT**

LOGOUT logs out the user from the database. The user may not use other commands except for QUIT once logout is completed. Below is an example where command format is ‘LOGOUT’

*Client: LOGOUT*

*Server: Received LOGOUT*

*Client: 200 OK*

**BALANCE**

BALANCE allows the client to view its current balance. Below is an example where the command format is 'BALANCE user\_id':

*client: BALANCE 3*

*server: Received: BALANCE 3*

*client: 200 OK*

*Balance for user Jane Smith: $10.00*

**QUIT**

QUIT terminates the client, after receiving the '200 OK' response from the server.

*client: QUIT*

*client: 200 OK*

**SHUTDOWN**

SHUTDOWN allows the client to terminate the connection and shut down the server. The server will respond with '200 OK' before it closes all sockets and database connection, and terminates.

*client: SHUTDOWN*

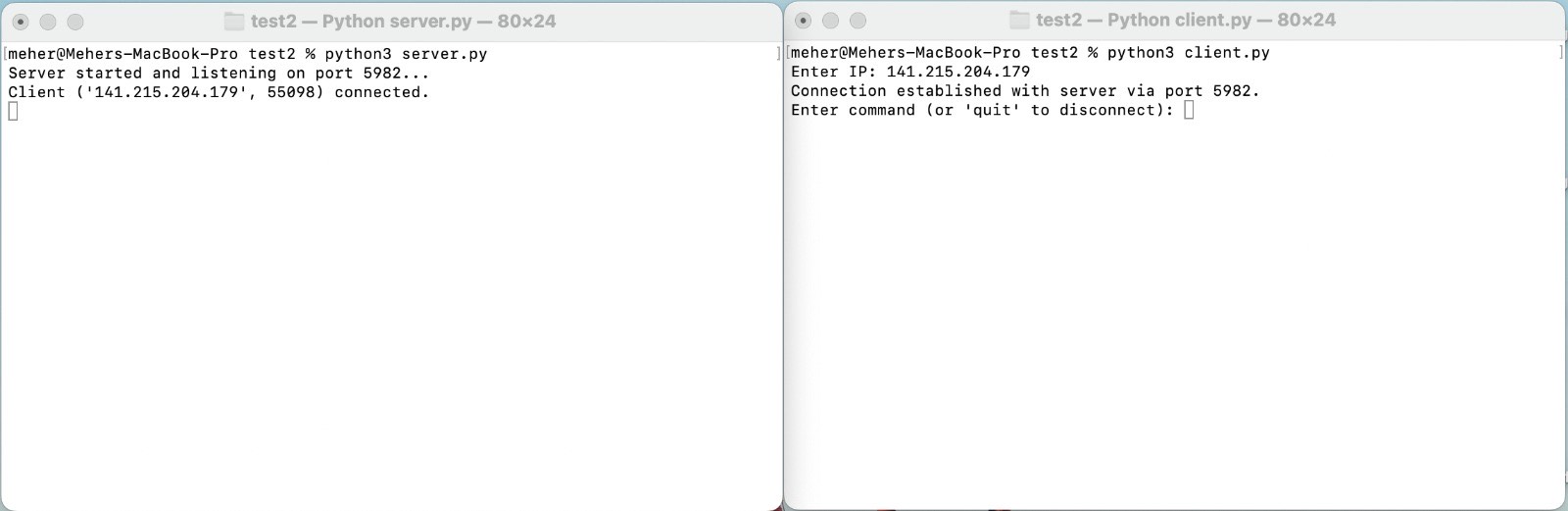
*server: Received: SHUTDOWN*

*client: 200 OK*

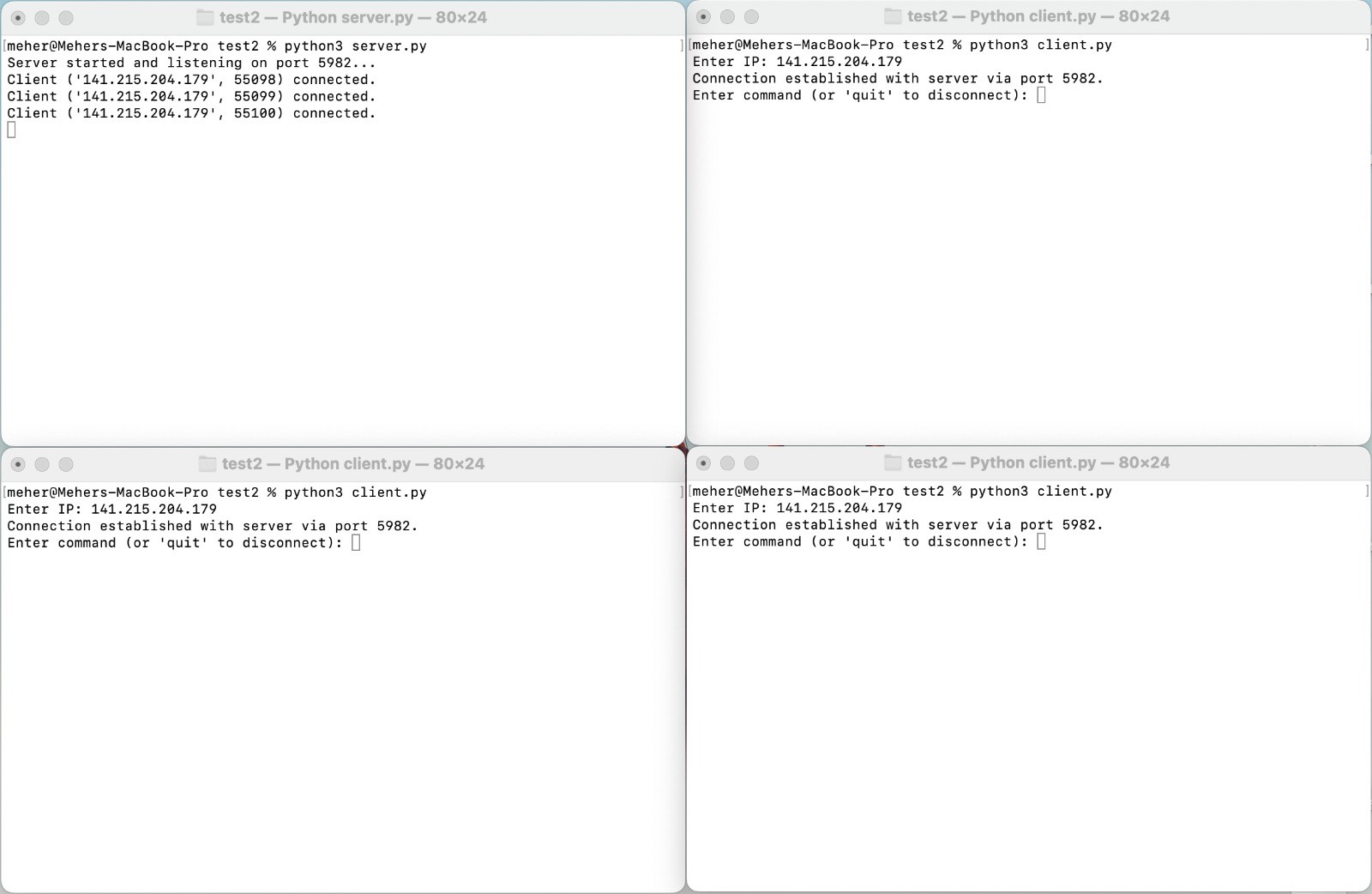
**Table with Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Description | Input Value | Expected output | Actual Output | Pass/Fail |
| 1 | Login test:  Correct username and password | Login Root password | 200 OK | 200 Ok | Pass |
| 2 | Login test:  Incorrect input | Login James random | 404 incorrect username or password | 404 incorrect username or password | Pass |
| 3 | LOGOUT | Logout | 200 OK | 200 OK | Pass |
| 4 | WHO | WHO | Root 127.0.0.1 | Root 127.0.0.1 | Pass |
| 5 | DEPOSIT | DEPOSIT 27.50 | New balance 127.50 | New balance 127.50 | Pass |
| 6 | LOOKUP | LOOKUP Char | Card Charizard and records | Card Charizard and records | Pass |
| 7 | DEPOSIT: incorrect integer input | DEPOSIT 32.786 | 404 invalid amount passed | 404 invalid amount passed | Pass |
| 8 | Who: non root user | WHO | 404 only root user may access who | 404 only root user may access who | Pass |
| 9 | Logout: attempt to use other commands | LOGOUT…  LIST | 404 must be logged in to access database | Allowed access to list | Fail |
| 10 | LOGOUT: attempt to use other commands | LOGOUT…  LIST | 404 must be logged in to access database | 404 must be logged in to access database | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Test Screenshot (Server and client 1)**

****

**Test Screenshot (server with client 1 and client 2)**

****

**Files Used**

* client.py - implements the client, initiates connection with server and sends requests to it
* server.py - implements the server, opens socket and listens for client connections and processes their request
* command\_handler.py - handles the processing of requests sent to the server
* constant.py - defines constants used in code (server port number, command names)
* database\_manager.py - controls interactions with the SQLite database and creates the database tables
* pokemon.db - SQLite database that stores all the data (cards inventory, user records)
* utilities.py - utility functions for processing the commands, response formatting, and error handling