**README for Online Pokémon Card Store Server and Client**

# Introduction

This README is to provide information about the Online Pokémon Card Store Server and its client program. The server is used to allow users to perform various actions related to buying and selling Pokémon cards. The instructions on implemented commands and instructions for running the server and client are listed below.

## Other Information

Platform: Cross-platform Programming Language: Python Student’s Roles:

* Josh K: README, Multiple Clients, Test Cases
* Jacob W: VIDEO, Functions and Threading, Testing

# Server

The server is the main control for managing the databases that store user and card information, as well as the user session table. The server is meant to receive commands from the client program. The server can handle 10 different clients at once before waiting to have one close to allow another client in.

## How to Run the Server

1. Open a terminal or command prompt.
2. Navigate to the project directory where server.py is stored.
3. Run the server file with the following command: **python server.py [host]**
   1. [host] is an optional argument that will be used to connect to the server host in the argument. If no argument is listed, then the local host IP will be used “127.0.01”.
4. The server will now be open and waiting for commands from the clients.

## Commands

### BUY

The ‘**BUY**’ command allows users to purchase Pokémon cards from the server. This command involves the following parameters:

* ‘card\_name’: The name of the Pokémon card.
* ‘card\_type’: The type of the Pokémon card.
* ‘card\_rarity’: The rarity of the Pokémon card.
* ‘card\_price’: The price of each card.
* ‘quantity’: The number of cards to purchase.
* ‘user\_id’: The ID of the user making the transaction.

When the command is executed, the server verifies the parameters, afterwards it checks the balance of the user. If successful, then the user’s balance is updated, and the cards are added to the user’s information, either adding it to the database or updating the existing one.

### SELL

The ‘**SELL**’ command allows users to sell Pokémon card to the server. This command uses the following parameters:

* ‘card\_name’: The name of the Pokémon card.
* ‘quantity’: The number of cards to purchase.
* ‘card\_price’: The price of each card.
* ‘user\_id’: The ID of the user making the transaction.

When the command is executed, the server verifies the parameters, afterwards it checks to make sure the number of cards are equal or less than the quantity of cards. If the conditions are met, the the card information is updated or removed if there are no more cards that the user owns, and the user’s balance is updated.

### LIST

The ‘**LIST**’ command allows users to see the list of cards owned by the user who sent the command. Only the root user is able to view all cards owned by everyone.

### BALANCE

The ‘**BALANCE**’ command allows the user to check the balance of a specific user ID. This command uses the following parameters:

* ‘user\_id’: The ID of the user whose balance will be shown.

If the user ID provided is valid, then it will return the specified user’s balance.

### LOGIN

The **‘LOGIN’** command allows the user to login to the server. Without being logged in, the user will only be able to user the QUIT command until they login.

### LOGOUT

The **‘LOGOUT’** command allows the user to logout of the server without closing the client. Once performed, the user will be only allowed to run the LOGIN or QUIT commands.

### DEPOSIT

The **‘DEPOSIT’** command allows a user to add money into their account. The command takes in a float as an argument. Once performed, it will update the user's account to properly add the money.

### WHO

The **‘WHO’** command is a command that is only allowed by the root user. It allows the root to see all users that are logged in and their IP address.

### LOOKUP

The **‘LOOKUP’** command allows the user to do a partial or full search for a card by entering it’s name, type, or rarity. If the cards exist in the users list, then it will return the results, otherwise it will a message saying no results found.

## Closing the Server

Ending the program can be done in two different means.

### QUIT

The QUIT command will end the client’s program, while keeping the server open for the next client program.

### SHUTDOWN

The SHUTDOWN command will end the server’s program. Only the root user will be able to run this command.

# Client

The client allows users to interact with the server by sending commands and receiving responses.

## How to Run the Client

1. Open the terminal or command prompt.
2. Navigate to the project directory where client.py is stored.
3. Run the client file with the following command: **python client.py [server\_host]**
   1. [server\_host] is an optional argument that will be used to connect to the server host in the argument. If no argument is listed, then the local host IP will be used “127.0.01”.
4. The client will attempt to connect to the server and prompt the user for commands. If the client is unable to connect to the server, then the program will end.

## Sending Commands

* Enter the desired command from the list from the server section in the client’s input, and the client will send the command to the server to be processed.
* To exit the client, send the ‘**QUIT’** command.

## Logging In

* While the server is running and the client is connected, Enter LOGIN with a valid Username and Password. If done successfully, you will be logged in and able to execute the other commands.

## Logging Out and QUIT

* When running the QUIT command, the user’s client will terminate, and the user will be logged out.
* When running LOGOUT, the user will still be connected to the server, but will only be able to run the LOGIN or QUIT commands.

# Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Function | Test Desc. | Input | Expected | Actual | Pass/Fail |
| 1 | LOGIN | Valid | LOGIN Root Root01 | 200 OK  User Root successfully signed in | 200 OK  User Root successfully signed in | Pass |
| 2 | LOGIN | Incorrect Username | LOGIN Test Root01 | 403 Wrong UserID or Password  User does not exist or password incorrect | 403 Wrong UserID or Password  User does not exist or password incorrect | Pass |
| 3 | LOGIN | Incorrect Password | LOGIN Root Test | 403 Wrong UserID or Password  User does not exist or password incorrect | 403 Wrong UserID or Password  User does not exist or password incorrect | Pass |
| 4 | DEPOSIT | Valid | DEPOSIT 100 | c: DEPOSIT 100  200 OK  Deposit successful. New User Balance $100000000200.00 | c: DEPOSIT 100  200 OK  Deposit successful. New User Balance $100000000200.00 | Pass |
| 5 | DEPOSIT | Non-float | DEPOSIT a | 405 message format order  DEPOSIT requires a float for money to add | 405 message format order  DEPOSIT requires a float for money to add | Pass |
| 6 | WHO | Client | WHO | 400 invalid command  Only user Root can execute this command | 400 invalid command  Only user Root can execute this command | Pass |
| 7 | WHO | Root | WHO | 200 OK  The list of active users:  Root 127.0.0.1 | 200 OK  The list of active users:  Root 127.0.0.1 | Pass |
| 8 | LOOKUP | Full | LOOKUP Pikachu | 200 OK  Found 2 match(es)  id card\_name card\_type rarity count  owner\_id  1 Pikachu Electric Common 3  1  4 Pikachu Electric Rare 6  1 | 200 OK  Found 2 match(es)  id card\_name card\_type rarity count  owner\_id  1 Pikachu Electric Common 3  1  4 Pikachu Electric Rare 6  1 | Pass |
| 9 | LOOKUP | Partial | LOOKUP Pik | 200 OK  Found 3 match(es)  id card\_name card\_type rarity count  owner\_id  1 Pikachu Electric Common 3  1  4 Pikachu Electric Rare 6  1  6 Pik Electric Rare 8  1 | 200 OK  Found 3 match(es)  id card\_name card\_type rarity count  owner\_id  1 Pikachu Electric Common 3  1  4 Pikachu Electric Rare 6  1  6 Pik Electric Rare 8  1 | Pass |
| 10 | LOOKUP | No Result | LOOKUP asdf | 404 no record found  Your search did not match any records | 404 no record found  Your search did not match any records | Pass |
| 11 | LIST | User | LIST | 200 OK  The list of records in the Pokemon cards table for current user, DefaultUser:  id card\_name card\_type rarity count  owner\_id  2 Diglet Electric Common 3  2  8 Pikachu Electric Common 1  2 | 200 OK  The list of records in the Pokemon cards table for current user, DefaultUser:  id card\_name card\_type rarity count  owner\_id  2 Diglet Electric Common 3  2  8 Pikachu Electric Common 1  2 | Pass |
| 12 | LIST | Root | LIST | The list of records in the Pokemon cards table:  id card\_name card\_type rarity count  owner\_id  1 Pikachu Electric Common 3  1  2 Diglet Electric Common 3  2  3 idgaf Electric Common 1  1  4 Pikachu Electric Rare 6  1  5 Pi Electric Rare 4  1  6 Pik Electric Rare 8  1  8 Pikachu Electric Common 1  2 | The list of records in the Pokemon cards table:  id card\_name card\_type rarity count  owner\_id  1 Pikachu Electric Common 3  1  2 Diglet Electric Common 3  2  3 idgaf Electric Common 1  1  4 Pikachu Electric Rare 6  1  5 Pi Electric Rare 4  1  6 Pik Electric Rare 8  1  8 Pikachu Electric Common 1  2 | Pass |
| 13 | LOGOUT | N/A |  | 200 OK | 200 OK | Pass |
| 14 | QUIT | N/A |  | 200 OK | 200 OK | Pass |
| 15 | SHUTDOWN | User |  | 401 security refusal  Not Root User | 401 security refusal  Not Root User | Pass |
| 16 | SHUTDOWN | Root |  | 200 OK  Shutting Down Server... | 200 OK  Shutting Down Server... | Pass |

Screenshots for two clients with 1 thread allowed

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Screenshots for two clients connected concurrently (10 threads allowed)

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated