CMPE 472_SEC 3

Computer Networks

Name: Mustafa Baran Ercan
No: 2881055520
Date of Submission: 03/11/2024

Table of Contents

1. Introduction

In this report, the design and implementation of a Python socket programming application for the "Stock Market Prediction" game is discussed. The program consists of a client that tries to estimate the stock's price within a tolerance range and a server that picks a stock at random from an Excel file called "stock.xlsx." The server either ends the connection if requested to or provides feedback on each guess, allowing up to three attempts per game. Using Python sockets on a local machine (localhost) on port 8888, this setup demonstrates the basic client-server communication model.

Example run of the program with correct guess at the end:

```
Successfully connected to the server.

Server: Predict the price for YKBNK
Enter your guess (or write END to Exit): 2500

Server: Higher! You have 2 tries left
Enter your guess (or write END to Exit): -90

Server: Invalid input. Please enter a non-negative numeric value.
Enter your guess (or write END to Exit):

Server: Please enter a valid number.
Enter your guess (or write END to Exit): 6000

Server: Lower! You have 1 tries left
Enter your guess (or write END to Exit): deneme

Server: Invalid input. Please enter a non-negative numeric value.
Enter your guess (or write END to Exit): 2750

Server: Success!
Game over. Bye Bye !
Client's connection is closed.
```

```
Client connected from address ('127.0.0.1', 49771)
Client guessed: -180
Client guessed: string input
Client guessed: ende
Client guessed: end
Client terminated the connection by sending END.
Client disconnected.
Server is closing down too...
Client connected from address ('127.0.0.1', 49782)
Client guessed: 2500
Client guessed: -90
Client guessed: EMPTY
Client entered empty input.
Client guessed: 6000
Client guessed: deneme
Client guessed: 2750
The client guessed correctly.
```

2. Explanation of *server.py*

2.1. Loading Stock Data and Random Stock Selection

The server uses the Pandas library for importing stock data from an Excel file (stock.xlsx). After importing the data, the server chooses a stock at random and delivers it to the client.

```python
import pandas as pd

# Load the stock data from the attached Excel file (stock.xlsx) on the server side
stocks_df = pd.read_excel("stock.xlsx")

def handle_request(client_conn):
    try:
        # Select a random stock
        selected_stock = stocks_df.sample().iloc[0] # sample() returns a random sample
        symbol = selected_stock['Stock Symbol']
        price = selected_stock['Price']  # int
```

2.2. Validating Guesses and Sending Feedback

The client is given three chances by the server to guess the stock price with a 5% tolerance. Depending on whether the guess was accurate, too high, or too low, the server provides feedback after every guess. The connection ends if the client requests to end the game or runs out of tries.

```python
while True:
    try:
        # Get the client's guess
        client_guess = client_conn.recv(1024).decode().strip()  # 1024: buffer size, decode(): bytes to string, strip(): remove whitespaces
        print("Client guessed: {0}".format(client_guess))

        if client_guess.upper() == "END": # Close connection
            print("Client terminated the connection by sending END.")
            client_conn.close() # Close the connection
            print("Client disconnected.\nServer is closing down too...")
            return # Exit the function

        # Handle empty input
        if client_guess == "EMPTY":
            print("Client entered empty input.")
            client_conn.send("Please enter a valid number.\n".encode())
            continue

        # Validate that the input is a non-negative number
        try:
            client_guess = float(client_guess)
            if client_guess < 0:
                raise ValueError
        except ValueError:
            client_conn.send("Invalid input. Please enter a non-negative numeric value.\n".encode())
            continue

        # Check if the guess is within 5% of the actual price
        if abs(client_guess - price) <= price * tolerance: # If is 5%
            client_conn.send("Success!\n".encode()) # Send a success message
            print("The client guessed correctly.")
            break

        elif client_guess < price:  # Higher
            tries += 1 # Increment the tries
            if tries >= 3:
                client_conn.send("Unlucky, the correct price was: {0}\n".format(str(price)).encode())
                print("The client spent 3 guesses incorrectly.")
                break
            else:
                client_conn.send(("Higher! You have " + str((3-tries)) + " tries left\n").encode())

        else:   # Lower
            tries += 1 # Increment the tries
            if tries >= 3:
                client_conn.send("Unlucky, the correct price was: {0}\n".format(str(price)).encode())
                print("The client spent 3 guesses incorrectly.")
                break
            else:
                client_conn.send(("Lower! You have " + str((3-tries)) + " tries left\n").encode())
```

1: Handles user input “END”

2: Handles empty user input (Ps: Client side has empty-input handler for transmission issues)

3: Handles other invalid inputs (negative numbers or strings)

4: Correct guesses

5: Higher price guess than the actual price

6: Lower price guess than the actual price

3. Explanation of client.py

3.1. Interactions & sending guesses

The client program asks the user to guess the stock price or exit the game by typing "END." Every guess is transmitted to the server, and the client gets provided with the response.

```python
# Get input from client
client_input = input("Enter your guess (or write END to Exit): ").strip()

if client_input.upper() == "END":
    client_socket.send(client_input.encode())
    print("Client ended the connection.")
    return

# Add placeholder for empty input (So the server can handle it)
if client_input == "":
    client_input = "EMPTY"

# Send the input to the server
client_socket.send(client_input.encode())
```

3.2. Handling Feedback and Terminating Conditions

The client decodes server messages for game-ending conditions, hints, or accurate guesses. Depending on user input or whether the server ends the game, the program ends the connection.

Server ends the game (Ran out of tries):

```python
# You either guessed correctly, or ran out of tries
if "Unlucky" in server_message or "Success" in server_message:
    print("Game over. Bye Bye !")
    break # Exit the loop
```

User ends the game by writing “END”:

```python
if client_input.upper() == "END":
    client_socket.send(client_input.encode())
    print("Client ended the connection.")
    return
```

4. Example Scenarios with Code Snippets

(The left side of the images are Client-side, and the right side of the images are Server-side)

4.1. Scenario 1: Successful Guess

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

PS C:\Users\musta\Desktop\Network> & c:/Users/musta/Desktop/Network/.venv/Scripts/python.exe c:/Users/
musta/Desktop/Network/client.py
Successfully connected to the server.

Server: Predict the price for ISCTR
Enter your guess (or write END to Exit): 1200

Server: Higher! You have 2 tries left
Enter your guess (or write END to Exit): 4500

Server: Lower! You have 1 tries left
Enter your guess (or write END to Exit): 3250

Server: Success!
Game over. Bye Bye !
Client's connection is closed.
```

```
PS C:\Users\musta\Desktop\Network> & c:/Users/musta/
musta/Desktop/Network/server.py
Server is running, waiting for connections...
Client connected from address ('127.0.0.1', 49714)
Client guessed: 1200
Client guessed: 4500
Client guessed: 3250
The client guessed correctly.
```

4.2. Scenario 2: Exceeding Guess Limit

```
PS C:\Users\musta\Desktop\Network> & c:/Users/musta/Desktop/Network
/.venv/Scripts/python.exe c:/Users/musta/Desktop/Network/client.py
Successfully connected to the server.

Server: Predict the price for GOOGL
Enter your guess (or write END to Exit): 500

Server: Higher! You have 2 tries left
Enter your guess (or write END to Exit): 2500

Server: Higher! You have 1 tries left
Enter your guess (or write END to Exit): 5000

Server: Unlucky, the correct price was: 10000
Game over. Bye Bye !
Client's connection is closed.
```

```
PS C:\Users\musta\Desktop\Network> & c:/Users/musta/Deskt
Server is running, waiting for connections...
Client connected from address ('127.0.0.1', 49734)
Client guessed: 500
Client guessed: 2500
Client guessed: 5000
The client spent 3 guesses incorrectly.
```

4.3. Scenario 3: Wrong Inputs & Ending the Game with Input "END"

```
PS C:\Users\musta\Desktop\Network> & c:/Users/musta/Desktop/Network/
.venv/Scripts/python.exe c:/Users/musta/Desktop/Network/client.py
Successfully connected to the server.

Server: Predict the price for YKBNK
Enter your guess (or write END to Exit): -180

Server: Invalid input. Please enter a non-negative numeric value.
Enter your guess (or write END to Exit): string input

Server: Invalid input. Please enter a non-negative numeric value.
Enter your guess (or write END to Exit): ende

Server: Invalid input. Please enter a non-negative numeric value.
Enter your guess (or write END to Exit): end
Client ended the connection.
Client's connection is closed.
```

```
PS C:\Users\musta\Desktop\Network> & c:/Users/musta/Desktop/Network/.v
Server is running, waiting for connections...
Client connected from address ('127.0.0.1', 49771)
Client guessed: -180
Client guessed: string input
Client guessed: ende
Client guessed: end
Client terminated the connection by sending END.
Client disconnected.
Server is closing down too...
```