University of Dhaka

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 2:

Introduction to Socket Programming
Exercises on Simple Client-Server Communication

Submitted By:

1. Syed Mumtahin Mahmud, Roll: 50

2. Nazira Jesmin Lina, Roll: 55

Submitted On :

January 24, 2023

Submitted To :

Dr. Md. Abdur Razzaque

Md Mahmudur Rahman

Md. Ashraful Islam

Md. Fahim Arefin

1 Introduction

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

The client-server model is a common architecture used in many networked systems, including the World Wide Web, email, and file sharing.

1.1 Objectives

- How application programs use protocol software to communicate acrossnetworks and internets.
- Introduction to Client-Server paradigm of interaction

2 Theory

Socket programs are used to communicate between various processes usually running on different systems. It is mostly used to create a client-server environment.Client-server communication refers to the process of exchanging information and data between a client and a server over a network. The client, also called the requesting device, initiates a request for a specific resource, service or information from the server, also known as the providing device. The server then processes the request and sends a response back to the client. The response may include the requested information, an acknowledgement of the request, or an error message if the request could not be fulfilled.

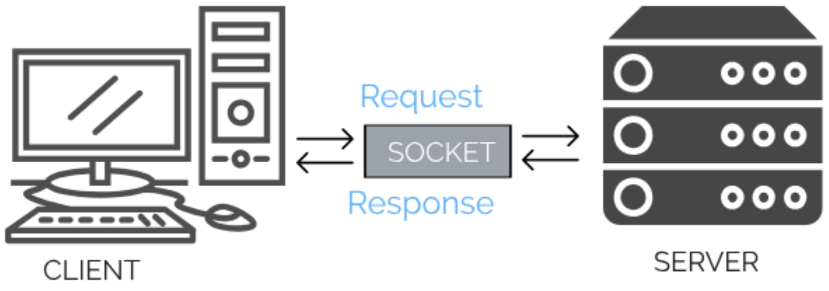


Figure 1: Client-Server model

The communication between the client and server typically follows a specific protocol, such as HTTP, TCP/IP, FTP etc. which defines the format and structure of the requests and responses. The client-server communication model is widely used in distributed computing systems and is the foundation for many networked applications, such as the World Wide Web, email, and file sharing. Simple client-server communication typically involves the following steps:

The client sends a request to the server over a network, using a specific protocol, such as HTTP or TCP/IP. The server receives the request and processes it, accessing any necessary resources or data. The server then sends a response back to the client, which may include requested data or a message indicating the status of the request. The client receives the response and processes it, displaying the data or taking further action as necessary.

3 Methodology

3.1 Server

A server is a computer or system that provides resources, data, or services to other computers or clients over a network.

On the server side when we turn it on it will wait for any client request. If it gets any request then it will establish a connection. After setting up the connection, it receives a query corresponding to which file client requested .In this section Our task was Client sends Small letter .Server receives it and converts it to Capital letter and checks whether a number is prime or not and sends the result to client.We also implemented a Bank server .When a client requests for checking balance,cash deposit, cash withdrawal server does the operation and send back the result to the client .It can be said that server accepts all valid requests and rejects the requests that are invalid.

3.2 Client

A client is a computer or system that requests and uses the resources, data, or services provided by a server.

Here our client side is any web browser .We will enter the IP address of our server and the port number. Then a request will be sent from client to the server. In our lab task we had to implement an ATM booth machine as a client. We will see some answers to the query that the server provided.

3.3 Problem A

Establishing a TCP connection in between a server process, running on host A and a client process, running on host B and then perform some operation by the server process requested by the client and send responses from the server.

1. Small letter to capital conversion for a line of text
2. Checking whether a number is prime or not

Server:

```
import socket
import sympy


def server_program():

    hostname=socket.gethostname()
    host=socket.gethostbyname(hostname)
    port = 4537
    print(host)

    server_socket = socket.socket()
    server_socket.bind(('', port))
    server_socket.listen(2)
    conn, address = server_socket.accept()
    print("Connection from: " + str(address))
    while True:
        data = conn.recv(1024).decode() #receiving messegee
        if not data:
            break
        if data.isdigit():
            print('number: '+data)
            if(sympy.isprime(int(data))): #checking if a number is prime
                conn.send(('its prime').encode())
            else:
                conn.send(('its not prime').encode())


        else:
            print("From connected user: " + str(data))
            data =data.upper() #convert string to uppercase letter
            conn.send(data.encode())  # send data to the client


    conn.close()  # close the connection


server_program()
```

Client:

```
import socket

def client_program():
    host = ''
    port = 4537

    client_socket = socket.socket()
    client_socket.connect((host, port)) #connecting with server

    message = input(" -> ")
    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())  #sending message
        data = client_socket.recv(1024).decode()  #receiving message
        print('Received from server: ' + data)
        message = input(" -> ")
    client_socket.close()
client_program()
```

3.4 Problem B

Create a TCP connection and design and implement a non-idempotent operation between bank server and ATM booth (client).

Server:

```
import socket
import sympy
import random


def server_program():
    user1='dipto'
    pass1='12345'
    blance1= 50000

    user2='lina'
    pass2='12345'
    pass2=20000

    dict = {}


    hostname=socket.gethostname()
    host=socket.gethostbyname(hostname)
    port = 4532
    print(host)

    server_socket = socket.socket()
    server_socket.bind(('', port))
    server_socket.listen(5)
    conn, address = server_socket.accept()


    print("Connection from: " + str(address))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break
        if data==user1:
            data = conn.recv(1024).decode()
            if data==pass1:
                conn.send(('40').encode())
                while True:
                    choice=conn.recv(1024).decode()
                    print('choice',choice)
                    if choice=='close':
                        return
                    if int(choice)==1: #choice 1 for checking balance
                        print('choice: ',choice)
                        conn.send(('Your Balance is: '+ str(blance1)).encode())
                    elif int(choice)==3: #choice3 for cash deposite
                        amnt=int(conn.recv(1024).decode())
                        blance1=blance1+amnt
                        conn.send(('New amount added. Your Balance is: '+

                        str(blance1)).encode())
                    else: # for cash withdrawal
                        amnt=int(conn.recv(1024).decode())
                        print(amnt)
                        if amnt<=0: # if requested amount is negative
                            continue
                        id=conn.recv(1024).decode()
                        if dict.get(id) is not None:
                            print("Error")
                            conn.send(('555').encode())

                        else:
                            print('withdrawn amount: ',amnt)
                            if amnt>blance1:#if requested amount is grater than acc balance
                                conn.send(('501').encode())
                                continue
                            else:

                                dict[id]={user1,amnt}
                                rand=random.randint(0,100)
                                print('rand',rand)
                                if (rand>70):
                                    conn.send(('555').encode())
                                    continue
                                blance1=blance1-amnt
                                conn.send(('Withdrawn successful. Your Balance is: '+
                                str(blance1)).encode())
                            print(dict)
            else:
                print('Invalid Password') # if password is not match
                conn.send(('404').encode())

        else:
            print('Invalid User') # if user id is not match
            conn.send(('404').encode())
    # conn.close()  # close the connection

server_program()
```

Client :

```
import socket
import time
import random


def send_withdraw_req(id,wit,client_socket):
    client_socket.send(wit.encode())
    time.sleep(1)
    strid=str(id)
    client_socket.send(strid.encode())

def client_program():
    host = ''
    port = 4532
    client_socket = socket.socket()
    client_socket.connect((host, port))
    id=0

    us=input('Enter your username: ')
    client_socket.send(us.encode())
    pas = input('Enter your password: ')
    client_socket.send(pas.encode())
    cc= client_socket.recv(1024).decode()
    if cc=='404':
        print('invalid Data')
        client_socket.close()
        return

    while True:


        print('Please Select')  # ATM booth menu option
        print('1.CHECK BALANCE')
        print('2.CASH WIHDRAWAL')
        print('3.CASH DEPOSITE')
        choose = input('ENTER : ') # inter desired option
        client_socket.send(choose.encode())
        if choose=='1':
            print(client_socket.recv(1024).decode())
        elif choose=='3':
            dep= input('Enter amount :') # enter deposite amount
            client_socket.send(dep.encode())
            print(client_socket.recv(1024).decode())
        elif choose=='2':
            id=id+1
            wit= input('Enter amount :') # enter withdrawal amount
            print(wit)
            wi=int(wit)
            if wi<=0:
                print('Invalid amount')
                client_socket.send(wit.encode())
            else :
                while True:
                    send_withdraw_req(id,wit,client_socket)

                    #send request id,withdrawal amount
                    rr=client_socket.recv(1024).decode()
                    print(rr)

                    if rr=='501':
                        print('Insufficient Balance')
                    elif rr=='555':
                        print("Transection Failed")   #after failing the transaction
                        print('Try Again?')
                        print('1.YES')
                        print('2.NO')
                        try_again= input('ENTER : ')
                        if try_again=='1':   #press 1 for again go back to the menu
                            client_socket.send(('2').encode())

                            #press 2 for closing connection
                            time.sleep(0.5)
                            continue
                        else:
                            break
                    else:
                        print(rr)
                        break
        print('Anything else') # after executing one request
        print('1.YES')
        print('2.NO')
        a= input('ENTER : ') # enter the desired option
        if a=='2':
            client_socket.send(('close').encode())
            break

    client_socket.close() #close the connection
client_program()
```

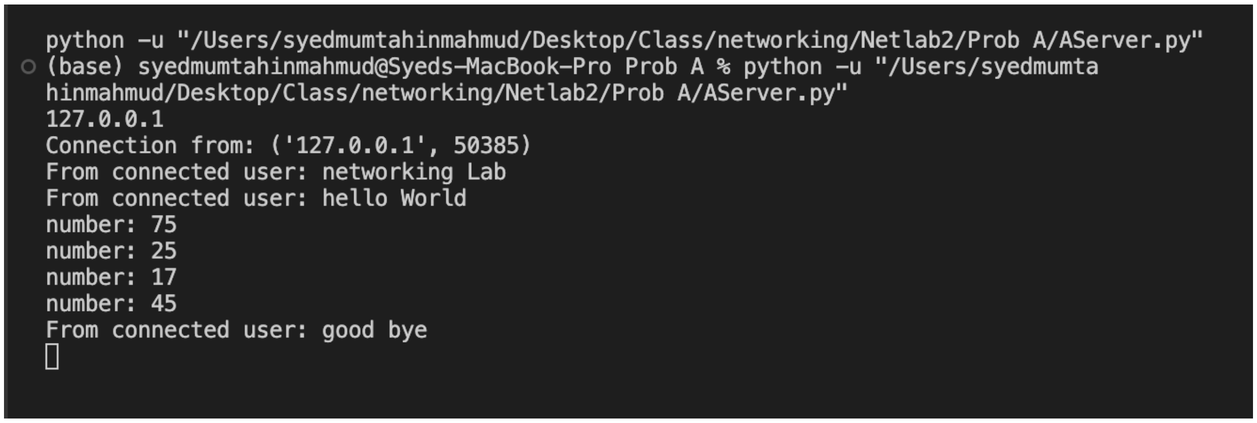
4 Experimental Result

Some Snapshots of the Client and Server Side queries can be seen in the following figures:

4.1 Problem A

Server :

Server takes a string or an integer from the client. It converts the string to an upper case string and send it back to the client. On the other hand, it check the integer if its a prime or not and send the result to the client.

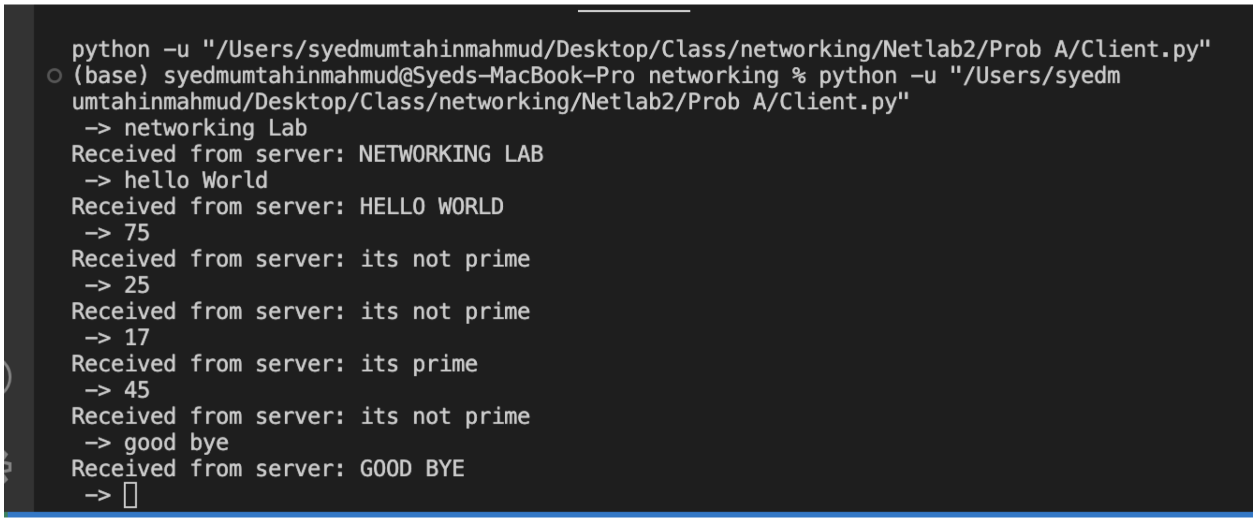


```
python -u "/Users/syedmumtahinmahmud/Desktop/Class/networking/Netlab2/Prob A/AServer.py"
(base) syedmumtahinmahmud@Syeds-MacBook-Pro Prob A % python -u "/Users/syedmumta
hinmahmud/Desktop/Class/networking/Netlab2/Prob A/AServer.py"
127.0.0.1
Connection from: ('127.0.0.1', 50385)
From connected user: networking Lab
From connected user: hello World
number: 75
number: 25
number: 17
number: 45
From connected user: good bye
```

Figure 2: Content of Server for Problem A

Client :

The client takes the input from the user and send it to the server. Then it takes the message form the server and show it in the console.



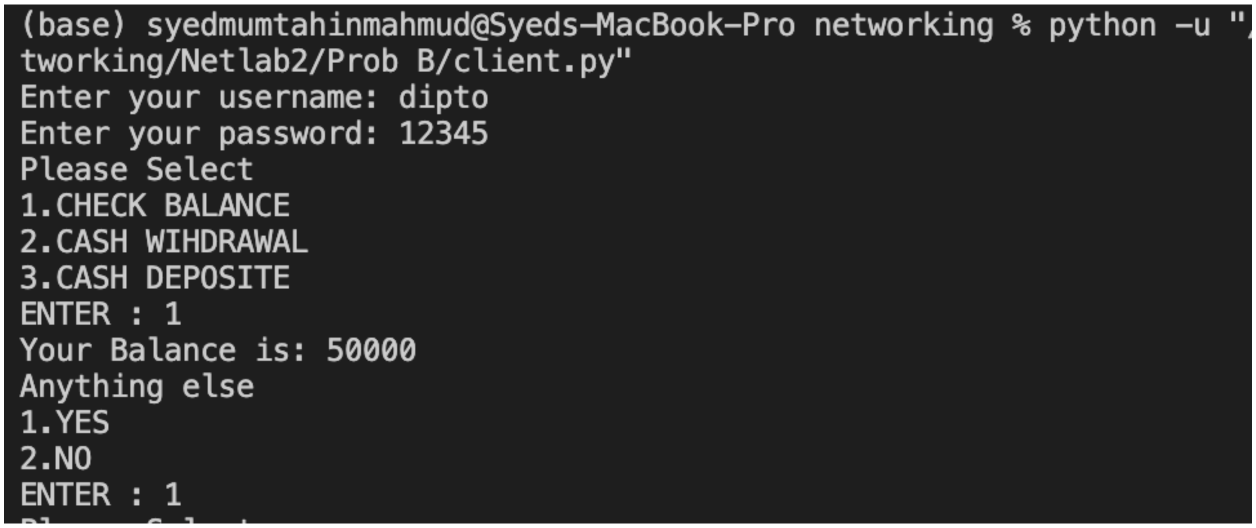
```
python -u "/Users/syedmumtahinmahmud/Desktop/Class/networking/Netlab2/Prob A/Client.py"
(base) syedmumtahinmahmud@Syeds-MacBook-Pro networking % python -u "/Users/syedm
umtahinmahmud/Desktop/Class/networking/Netlab2/Prob A/Client.py"
 -> networking Lab
Received from server: NETWORKING LAB
 -> hello World
Received from server: HELLO WORLD
 -> 75
Received from server: its not prime
 -> 25
Received from server: its not prime
 -> 17
Received from server: its prime
 -> 45
Received from server: its not prime
 -> good bye
Received from server: GOOD BYE
 ->
```

Figure 3: Content of Client for Problem A

4.2 Problem B

Balance Checking :

Here from the menu, users can take the option 1 for checking their current balance. After sending the option to the server, server will return the balance of the account.

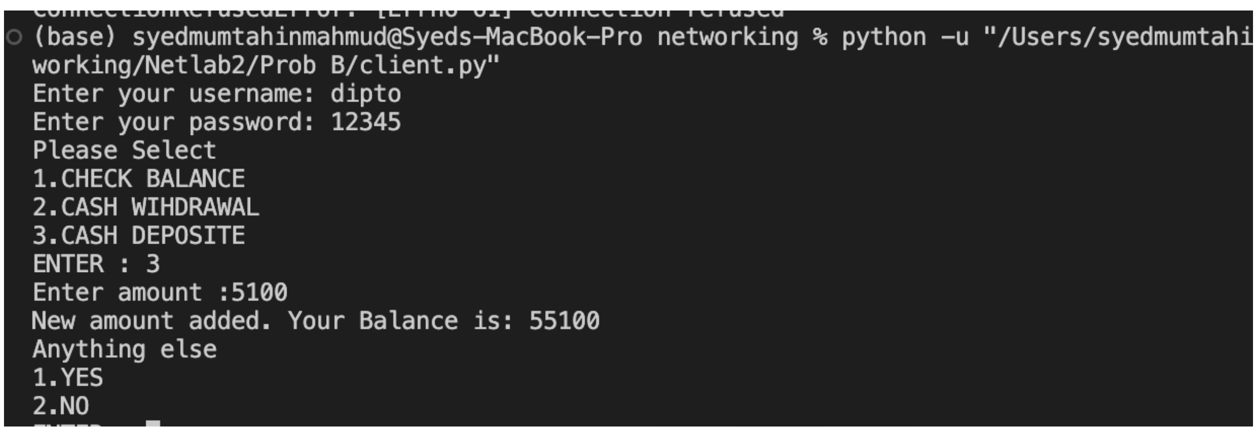


```
(base) syedmumtahinmahmud@Syeds-MacBook-Pro networking % python -u "
tworking/Netlab2/Prob B/client.py"
Enter your username: dipto
Enter your password: 12345
Please Select
1.CHECK BALANCE
2.CASH WIHDRAWAL
3.CASH DEPOSITE
ENTER : 1
Your Balance is: 50000
Anything else
1.YES
2.NO
ENTER : 1
```

Figure 4: Content of Check balance option

Cash Deposit :

By selecting this option, users will be able to deposit money to their account. After selecting this option, user will have to select the amount they wish to deposit and press enter. Server will add the amount with the existing balance and return the current balance of the account.

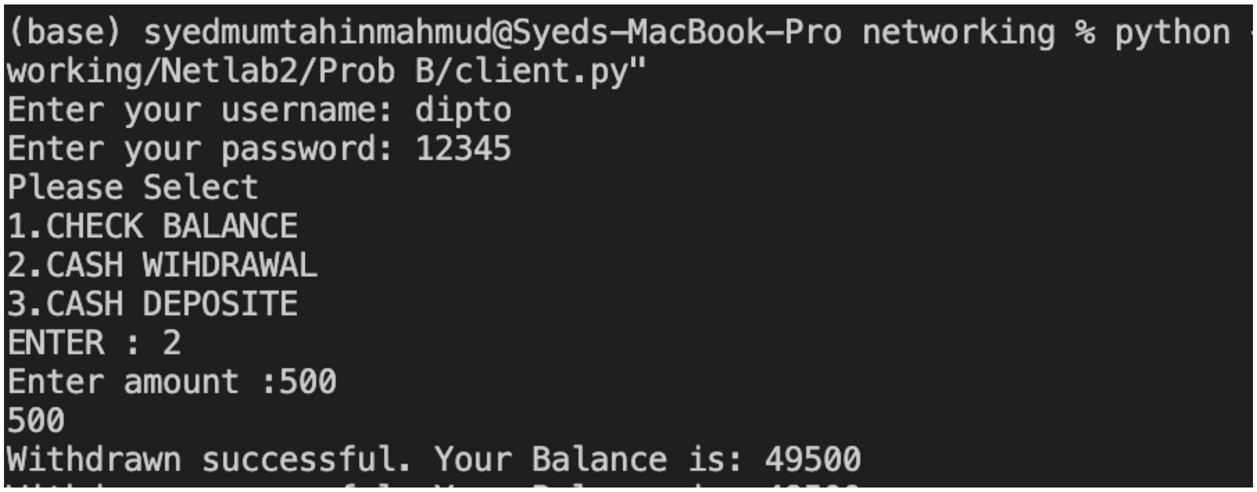


```
○ (base) syedmumtahinmahmud@Syeds-MacBook-Pro networking % python -u "/Users/syedmumtahi
  working/Netlab2/Prob B/client.py"
  Enter your username: dipto
  Enter your password: 12345
  Please Select
  1.CHECK BALANCE
  2.CASH WIHDRAWAL
  3.CASH DEPOSITE
  ENTER : 3
  Enter amount :5100
  New amount added. Your Balance is: 55100
  Anything else
  1.YES
  2.NO
```

Figure 5: Content of Cash deposit option

Cash Withdrawal :

User can withdraw money from their account by selecting the second option from the account. They will have to enter the money they want to withdraw and press enter.

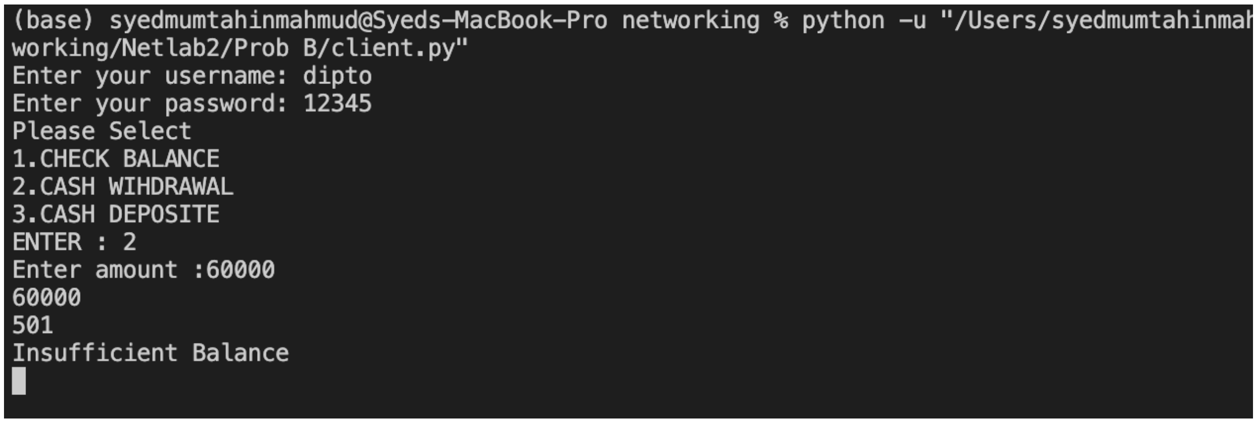


```
(base) syedmumtahinmahmud@Syeds-MacBook-Pro networking % python
working/Netlab2/Prob B/client.py"
Enter your username: dipto
Enter your password: 12345
Please Select
1.CHECK BALANCE
2.CASH WIHDRAWAL
3.CASH DEPOSITE
ENTER : 2
Enter amount :500
500
Withdrawn successful. Your Balance is: 49500
```

Figure 6: Content of Cash Withdrawal option

Insufficient Balance :

If a user want to withdraw more money than has in his account, then this message will be send form the server.

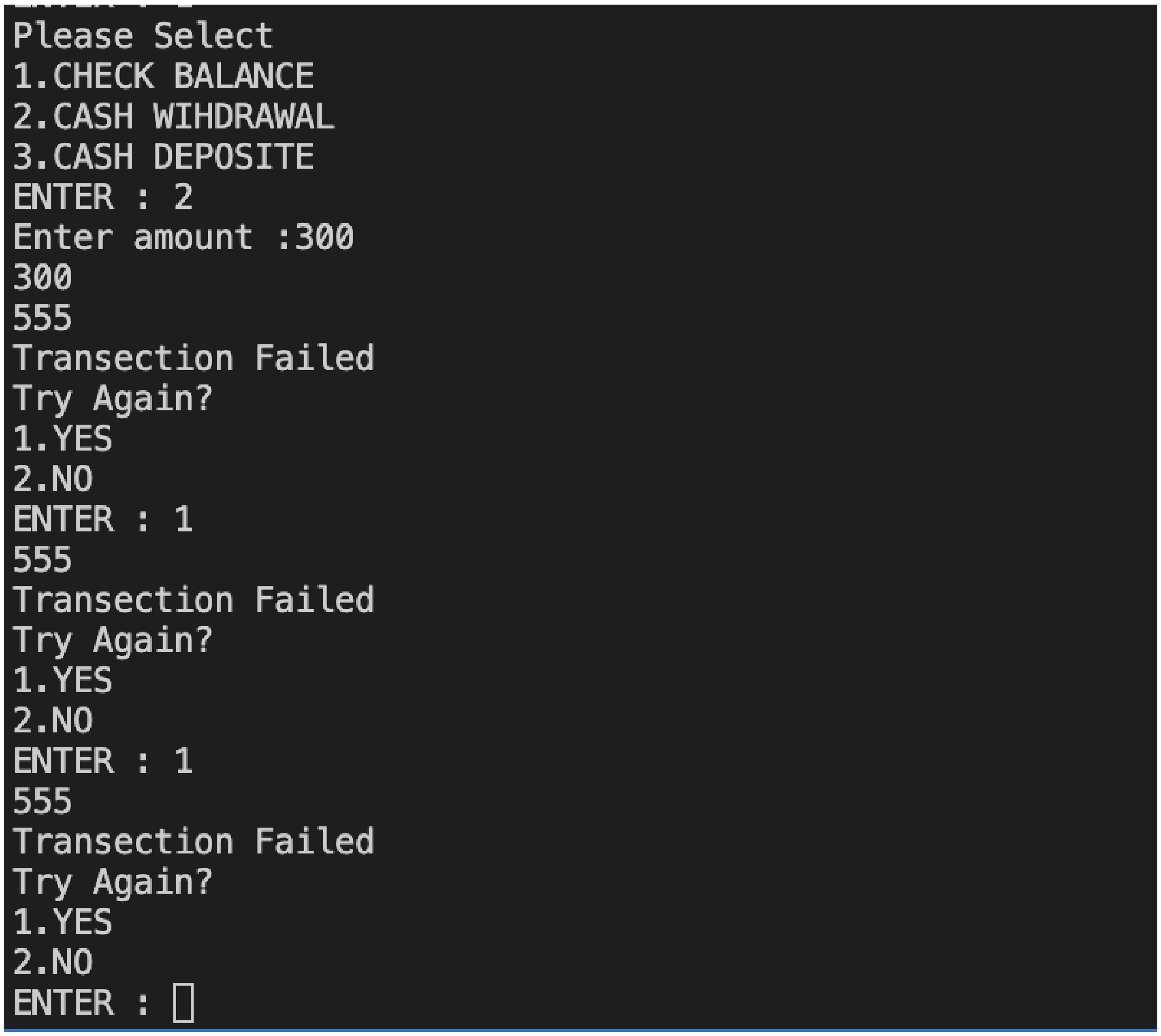


```
(base) syedmumtahinmahmud@Syeds-MacBook-Pro networking % python -u "/Users/syedmumtahinmah
working/Netlab2/Prob B/client.py"
Enter your username: dipto
Enter your password: 12345
Please Select
1.CHECK BALANCE
2.CASH WIHDRAWAL
3.CASH DEPOSITE
ENTER : 2
Enter amount :60000
60000
501
Insufficient Balance
```

Figure 7: Content of Client for Insufficient Balance

Error :

If a user puts an request for cash withdrawal, the request is saved using the request id ,user name, and amount. If for any circumstances the request process is failed because of re appearing of request, the server reject the request of the client. So , if a client resend requests by mistake or for any kind off network problem, if the money is withdrawn with the same request, the server will reject all the other requests.

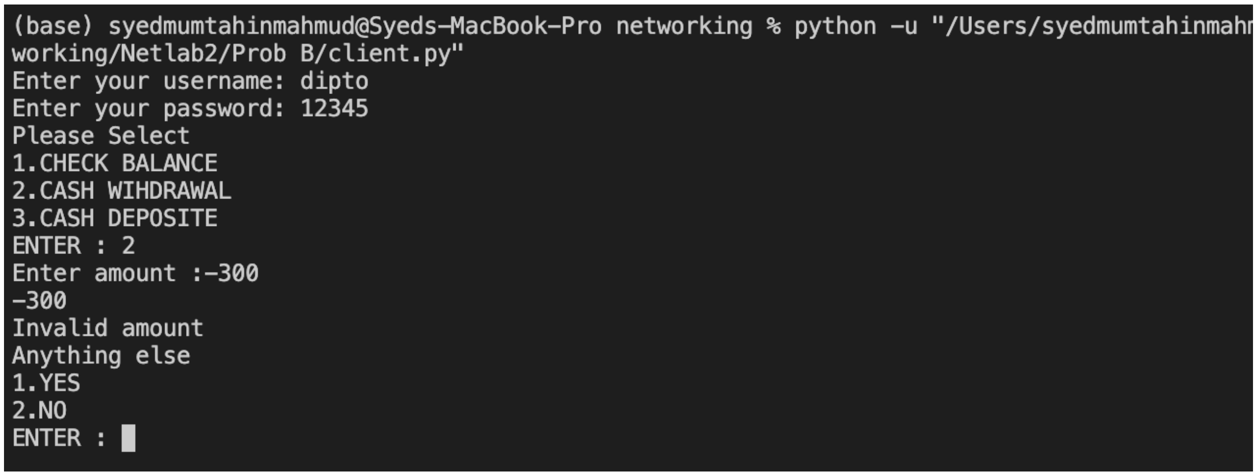


```
Please Select
1.CHECK BALANCE
2.CASH WIHDRAWAL
3.CASH DEPOSITE
ENTER : 2
Enter amount :300
300
555
Transection Failed
Try Again?
1.YES
2.NO
ENTER : 1
555
Transection Failed
Try Again?
1.YES
2.NO
ENTER : 1
555
Transection Failed
Try Again?
1.YES
2.NO
ENTER : 
```

Figure 8: Error Checking

Negative Balance :

If a user want to withdraw less than 00 from his account, then this message will be send form the server.



```
(base) syedmumtahinmahmud@Syeds-MacBook-Pro networking % python -u "/Users/syedmumtahinmahm
working/Netlab2/Prob B/client.py"
Enter your username: dipto
Enter your password: 12345
Please Select
1.CHECK BALANCE
2.CASH WIHDRAWAL
3.CASH DEPOSITE
ENTER : 2
Enter amount :-300
-300
Invalid amount
Anything else
1.YES
2.NO
ENTER :
```

Figure 9: Content of Client for entering negative balance

5 Experience

1. We had a wonderful experience with how servers and clients work in real life.
2. Experienced how socket programming work and establish connection between client and server.

References

[1] Socket programming in python. *GeeksforGeeks*, jun 20 2017. [Online; accessed 2023-01-25].

[2] Pankaj. Python socket programming - Server, client example. *DigitalOcean*, aug 3 2022. [Online; accessed 2023-01-25].

[3] Real Python. Socket programming in python (guide). *Real Python*, feb 21 2022. [Online; accessed 2023-01-25].