Assignment 8

Implementation of TCP/UDP Socket Programming

NAME: Shirish Manoj Bobde

Reg. No.: 812

Roll No.: ECE/21152

Problem Statement

Implement a client-server program using TCP/UDP sockets in Python for handling multiple clients on the server with multithreading. Your program should allow multiple clients to connect to the server simultaneously and exchange messages. Each client connection should be handled in a separate thread.

Code:

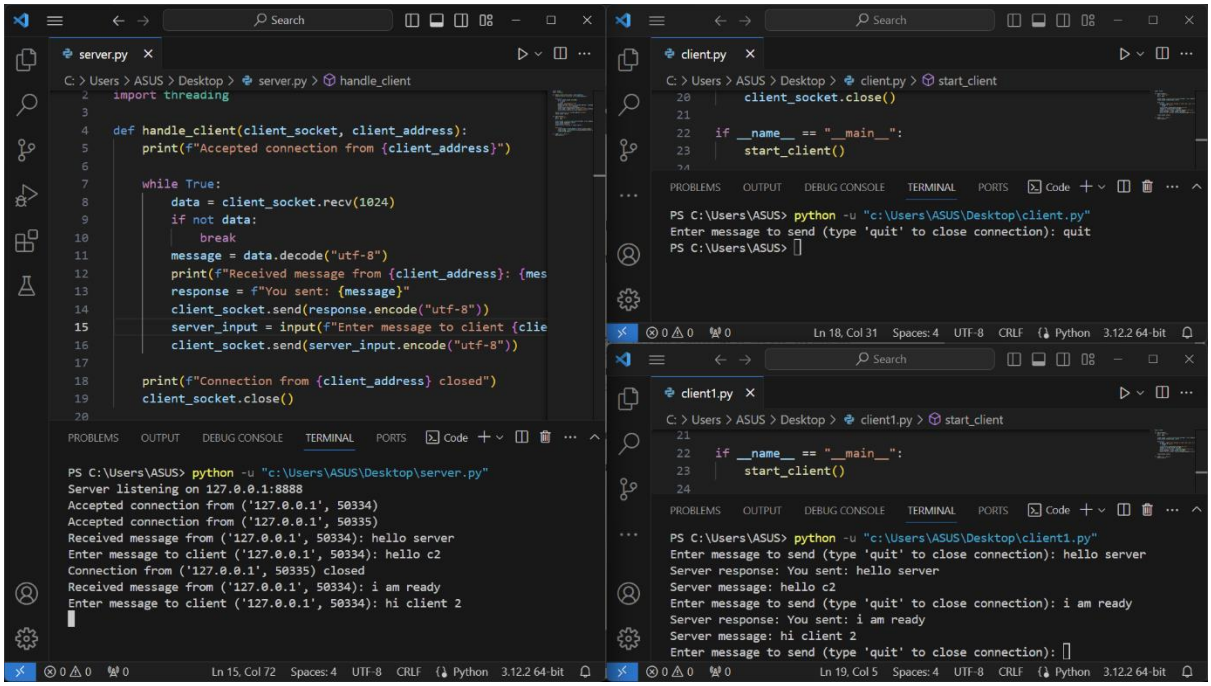
Client

```
import socket

def start_client():
    host = "127.0.0.1"
    port = 8888

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))

    while True:
        message = input("Enter message to send (type 'quit' to close
connection): ")
        if message == "quit":
            break
        client_socket.send(message.encode("utf-8"))
        response = client_socket.recv(1024)
        print(f"Server response: {response.decode('utf-8')}")
        server_message = client_socket.recv(1024)
        print(f"Server message: {server_message.decode('utf-8')}")

    client_socket.close()

if __name__ == "__main__":
    start_client()
```

Server

```
import socket
import threading

def handle_client(client_socket, client_address):
    print(f"Accepted connection from {client_address}")

    while True:
        data = client_socket.recv(1024)
        if not data:
            break
        message = data.decode("utf-8")
        print(f"Received message from {client_address}: {message}")
        response = f"You sent: {message}"
        client_socket.send(response.encode("utf-8"))
        server_input = input(f"Enter message to client {client_address}: ")
        client_socket.send(server_input.encode("utf-8"))

    print(f"Connection from {client_address} closed")
    client_socket.close()

def start_server():
    host = "127.0.0.1"
    port = 8888

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen(5)
    print(f"Server listening on {host}:{port}")

    while True:
        client_socket, client_address = server_socket.accept()
        client_thread = threading.Thread(target=handle_client,
args=(client_socket, client_address))
        client_thread.start()

if __name__ == "__main__":
    start_server()
```

Output


server.py
import threading
def handle_client(client_socket, client_address):
print(f"Accepted connection from {client_address}")
while True:
data = client_socket.recv(1024)
if not data:
break
message = data.decode("utf-8")
print(f"Received message from {client_address}: {mes
response = f"You sent: {message}"
client_socket.send(response.encode("utf-8"))
server_input = input(f"Enter message to client {clie
client_socket.send(server_input.encode("utf-8"))
print(f"Connection from {client_address} closed")
client_socket.close()
PS C:\Users\ASUS> python -u "c:\Users\ASUS\Desktop\server.py"
Server listening on 127.0.0.1:8888
Accepted connection from ('127.0.0.1', 50334)
Accepted connection from ('127.0.0.1', 50335)
Received message from ('127.0.0.1', 50334): hello server
Enter message to client ('127.0.0.1', 50334): hello c2
Connection from ('127.0.0.1', 50335) closed
Received message from ('127.0.0.1', 50334): i am ready
Enter message to client ('127.0.0.1', 50334): hi client 2
client.py
client_socket.close()
if __name__ == "__main__":
start_client()
PS C:\Users\ASUS> python -u "c:\Users\ASUS\Desktop\client.py"
Enter message to send (type 'quit' to close connection): quit
PS C:\Users\ASUS>
client1.py
if __name__ == "__main__":
start_client()
PS C:\Users\ASUS> python -u "c:\Users\ASUS\Desktop\client1.py"
Enter message to send (type 'quit' to close connection): hello server
Server response: You sent: hello server
Server message: hello c2
Enter message to send (type 'quit' to close connection): i am ready
Server response: You sent: i am ready
Server message: hi client 2
Enter message to send (type 'quit' to close connection):