



**Ganpat
University**

॥ विद्यया समाजोत्कर्षः ॥

**Institute of
Computer
Technology**

Name: Tushar Panchal

En.No: 21162101014

Sub: CN (Computer Networks)

Branch: CBA

Batch:51

PRACTICAL 07

❖ **AIM :** Socket Programming.

❖ **Scenario :**

An organization named Albert Enterprise has established two departments for better performance of the company, as each department will be having some specific set of tasks to perform. So, this will reduce the time and increase the efficiency of the work. As both the departments are dependent on each other, they need to communicate more frequently. To solve the problem, the IT department has suggested the option to create a chat application which will work only in the office premises. So, help the IT professionals to create the chat application.

Make sure that the application has the below mentioned features:

- 1) Department 1 will be set as the server while department 2 will be set as a client device.
- 2) If any of the device irrespective of client or server has sent the message that the “work is done”, then connection should be closed on both the ends.
- 3) Client device can send multiple messages at the same time, while the server device can send a single message at a time.
- 4) There is no restriction on the protocol selection, you can use UDP or TCP. Justify the reason for selection of the specific protocol.

✓ Source Code(prac7.py) :

```
import socket

def chat_server(server_ip, server_port):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_ip, server_port))
    server_socket.listen(1) # Listen for incoming connections

    print("Server is waiting for a connection...")
    client_socket, client_addr = server_socket.accept()
    print(f"Connected to {client_addr}")

    while True:
        message = input("Server: ")
        client_socket.send(message.encode())

        received_message = client_socket.recv(1024).decode()
        print(f"Client: {received_message}")

        if "work is done" in received_message.lower():
            break

    server_socket.close()
    client_socket.close()

def chat_client(server_ip, server_port):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((server_ip, server_port))

    while True:
        message = input("Client: ")
        client_socket.send(message.encode())

        received_message = client_socket.recv(1024).decode()
        print(f"Server: {received_message}")

        if "work is done" in received_message.lower():
            break

    client_socket.close()

def main():
    role = input("Enter 'server' or 'client': ")
    server_ip = '172.31.112.1'
    server_port = 1077

    if role.lower() == 'server':
        chat_server(server_ip, server_port)
```

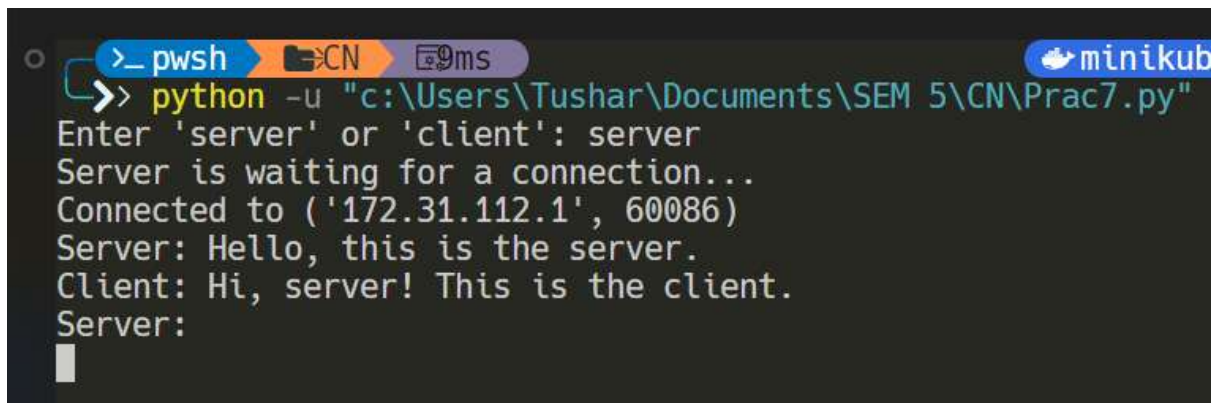
```

elif role.lower() == 'client':
    chat_client(server_ip, server_port)
else:
    print("Invalid role. Please enter 'server' or 'client'.")

if __name__ == "__main__":
    main()

```

✓ **Output (Server) :**

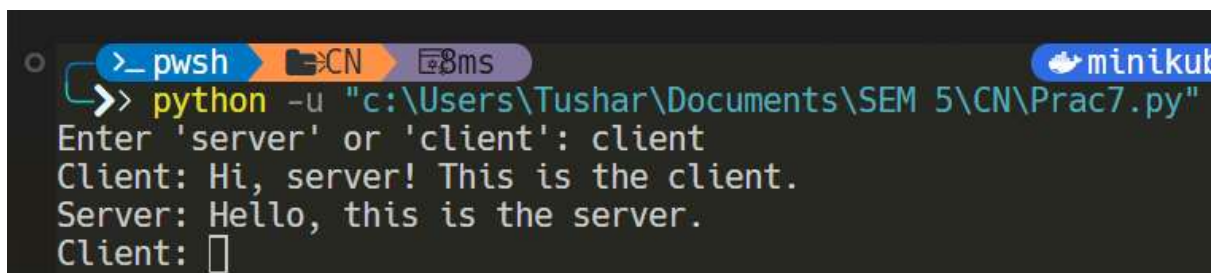


```

>_ pwsh CN 9ms minikub
>> python -u "c:\Users\Tushar\Documents\SEM 5\CN\Prac7.py"
Enter 'server' or 'client': server
Server is waiting for a connection...
Connected to ('172.31.112.1', 60086)
Server: Hello, this is the server.
Client: Hi, server! This is the client.
Server:

```

✓ **Output (Client) :**



```

>_ pwsh CN 8ms minikub
>> python -u "c:\Users\Tushar\Documents\SEM 5\CN\Prac7.py"
Enter 'server' or 'client': client
Client: Hi, server! This is the client.
Server: Hello, this is the server.
Client:

```

Conclusion :

this practical exercise provided a hands-on introduction to fundamental concepts in computer networks and socket programming. It gave you the opportunity to create a simple chat application and gain valuable experience in network communication, IP addressing, and port configuration. Understanding these principles is essential for building more complex networked applications and services.