

Exp NO : 12B

Date : 28.9.26

Implementation of chat client
server using TCP / UDP sockets

AIM :

To implement chat client server using the TCP and UDP sockets

Server side Algorithm:

import socket :

```
server = socket.socket(socket.AF_INET,  
                      socket.SOCK_STREAM)
```

```
server.bind(("localhost", 12345))
```

```
server.listen(1)
```

```
print("Server is waiting for connection...")
```

```
conn, addr = server.accept()
```

```
print(f"Connected to {addr}")
```

```
while True:
```

```
    msg = conn.recv(1024).decode()
```

```
    if msg.lower() == 'bye':
```

```
        print("Client disconnected")
```

```
        break
```

```
    print(f"Client : {msg}")
```

```
reply = input("you: ")  
com.send(reply.encode())  
if reply.lower() == 'bye':  
    break  
com.close()
```

Client side Algorithm:

```
import socket  
client = socket.socket(socket.AF_INET,  
                      socket.SOCK_STREAM)  
client.connect(('localhost', 12345))  
while True:  
    message = input("you: ")  
    client.send(message.encode())  
    if message.lower() == "bye":  
        break  
    reply = client.recv(1024).decode()  
    print(f"server {reply}")  
    if reply.lower() == 'bye':  
        break
```

client.close()

~~Sample Input and Output.~~

~~Server side :-~~

\$ server waiting for connection

connected to ('127.0.0.1', 59010)

Client : Hi server!

You : Hello Client!

Client : How are you?
You : I'm fine thanks!
Client : bye
Client disconnected

client side :
you : the server!
server : Hello client,
you : How are you?
server : I'm fine, thanks.
you : bye!

(Client -> Server) user -> kernel -> filter
(Server -> User) kernel -> user

Result
the server listens for UDP messages and displays them along with the sender's details
~~and also prints the message~~
(Client -> Server) at behavioral level