EXP NO : 12) b) chat client server.

Aim:
TO implement chat client server using TCP/UDP Packet sockets.

Source code:

TCP server.py:

```
import socket
import threading.
def handle_receive(client_socket):
    while True:
        try:
            message= client_socket. recv(1024).decode
            if message:
                print ("client :", message)
            else:
                break
        except:
            break
def handle_send (client_socket):
    while True:
        message= input ("server : ").
        client_socket.send (message. encode())
server_socket= socket. socket (socket. AF_INET,
                    socket. SOCK_STREAM)
server_socket. bind (("localhost", 5555))
server_socket. listen(1)
print ("server is waiting for connection...")
client_socket, addr = server_socket. accept()
print (f"connected by {addr}")
```

```
receive_thre
    handle
send_ threa
    handle_s
receive_threa
send_threa
receive_ thre
send_ thre
client_sock
server_soc

TCP client

import s
import
def hand
    while
        try
        me

        ex

def hand
    while
        me
        cli
client_sock

client_soc
```

```
t server.

ent server using

socket):

et. recv(1024).decode()

message)

ket):

er: ").
essage. encode())
ocket. AF_INET,
EAM)
t", 5555))

connection...")
ket.accept()
```

```
receive_thread = threading.Thread(target=
        handle_receive, args=(client_socket,))
send_thread = threading.Thread(target=
        handle_send, args=(client_socket))
receive_thread.start()
send_thread.start()
receive_thread.join()
send_thread.join()
client_socket.close()
server_socket.close()
```

TCP client.py:

```
import socket
import threading
def handle_receive(client_socket):
    while True:
        try:
            message= client_socket.recv(1024).decode()
            if message:
                print("server:", message)
            else:
                break
        except:
            break
def handle_send(client_socket):
    while True:
        message= input("client: ");
        client_socket.send(message.encode())
client_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_STREAM)
client_socket.connect(("localhost", 5555))
print("Connected to the server")
```

```
receive_thread = threading.Thread(target=
        handle_receive, args=(client_socket,))
send_thread = threading.Thread(target=
        handle_send, args=(client_socket))
receive_thread.start()
send_thread.start()
receive_thread.join()
send_thread.join()
client_socket.close().
```

output:

server.py output:

```
server is waiting for connection....
connected by ('127.0.0.1' <client_port>)
server: Hello, client!
client: Hi, server!
server: How are you?
client: I am good, thanks!
```

client.py output:

```
connected to the server
client: Hi, server!
server: Hello, client!
client: I am good, thanks!
server: How are you?
```

Result:

Thus, the program for chat client server using TCP/UDP was successfully executed and the output verified.

cket_callback,
r=0)

packet sniffing

Exp No : 15 Webalizer Tool

Aim:

To analyze the different types of web logs using Webalizer tool.

Procedure:

1. Run webalizer window version
2. Input web log file (down load from web)
3. Press run webalizer

Thus, the different types of web logs using Webalizer tool was successfully analyzed.