

Ex-12b)

Date: 23/10/24

AIM:

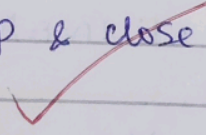
Implement chat client server using TCP/UDP sockets.

Algorithm:

chat-server.py :

- Create a UDP socket & bind it to 127.0.0.1 and port 12345.
- Wait for a message from a client using `recvfrom`.
- Decode & print the received message.
- Take input from the user, encode it, & send it back to the client using `sendto`.
- If the reply is 'end', break the loop & close the socket.

Recv2.py :

- Create a UDP socket.
 - Prompt the user for input & send the message.
 - Wait for a response.
 - Decode & print the message.
 - If the user input is 'end', break the loop & close the socket.
- 

code :

chat-server.py :

import socket

def recvr1():

port = 12345

host = '127.0.0.1'

with socket.socket(socket.AF_INET,
socket.SOCK_DGRAM) as s:

s.bind((host, port))

while True:

d, add = s.recvfrom(1024)

print("Client", {d.decode()})

a = input("Enter reply ")

s.sendto(a.encode(), add)

if a == "end":

break

exit.

recvr1()

recvr1.py :

import socket

import time

def recvr2(a):

host = '127.0.0.1'

port = 12345

with

socket.socket(socket.AF_INET, socket.
SOCK_DGRAM) as s:

s.sendto(a.encode(), (host, port))

d, add = s.recvfrom(1024)

print({d.decode()})


```
while(True):
```

```
    a = input("Enter message")
```

```
    if(a == "end"):
```

```
        recvr2(a)
```

```
        break
```

```
    else:
```

```
        recvr2(a)
```

OUTPUT: chat_servr.py:

```
> cd desktop
```

```
> python chat_servr.py
```

```
Client {'hello'}
```

```
Enter reply hi
```

```
Client {'I'm from REC'}
```

```
Enter Reply Me also.
```

recvr2.py:

```
> cd desktop
```

```
> python recvr2.py
```

```
Enter message hello
```

```
{'hi'}
```

```
Enter message I'm from REC
```

```
{'Me', 'also'}
```

```
Enter message
```

