DATE : 07/06/2021

ROLL NO : 1

EXPERIMENT NO - 8

MULTI-USER CHAT SERVER

AIM : To write a program to Implement Multi-user chat server using

TCP as transport layer protocol.

THEORY:

Sockets can be thought of as endpoints in a communication channel that is bidirectional, and establishes communication between a server and one or more clients. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server side. Any client that has a socket associated with the same port can communicate with the server socket.

The chat server does the following things:

1. Accept multiple incoming connections for client.
2. Read incoming messages from each client and broadcast them to all other

connected clients.

The chat client does the following 2 things :

1. Listen for incoming messages from the server.
2. Check user input. If the user types in a message then send it to the server.

Algorithm:

Server

1. Start
2. Import modules socket and select
3. Initialize the dictionary record
4. Create a socket named server_socket using socket function
5. Bind the server_socket with localhostand port
6. Listen for connection
7. The client id is received and checks for duplication
8. If client id is unique it is appended to the dictionary
9. If terminating message is entered the record is deleted from dictionary and remove from socket
10. Stop

Client

1. Start
2. Import modules socket, select, string, sys
3. Create a socket s for client using socket function
4. S is bonded to host and port
5. Messages are send to other clients via send socket function
6. Messages from server are received via recv() function
7. Stop

PROGRAM:

Server

```
import socket, select
def send_to_all (sock, message):
        for socket in connected_list:
                if socket != server_socket and socket != sock :
                        try :
                                socket.send(message)
                        except :
                                socket.close()
        connected_list.remove(socket)
if _name_ == "_main_":
        name=""
        record={}
        connected_list = []
        buffer = 4096
        port = 5001
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.bind(("localhost", port))
        server_socket.listen(10)
        connected_list.append(server_socket)
        print ("\t\t\t\t**SERVER WORKING**\n\n ")
        while True:
                rList,wList,error_sockets = select.select(connected_list,[],[])
                for sock in rList:
                        if sock == server_socket:
                                sockfd, addr = server_socket.accept()
                                name=sockfd.recv(buffer)
                                connected_list.append(sockfd)
                                record[addr]=""
```

```
                    if name in record.values():
                            sockfd.send("\r Username already taken!\n")
                            del record[addr]
                            connected_list.remove(sockfd)
                            sockfd.close()
                            continue
                    else:
                            record[addr]=name
                            print "Client (%s, %s) connected" % addr," [",record[addr],"]"
                    sockfd.send("\r..... Welcome to the chat. Enter 'bye' anytime to
                    exit....\n\n") send_to_all(sockfd, "\r "+name+" joined the conversation
                    \n")
            else:
                    try:
                            data1 = sock.recv(buffer)
                            data=data1[:data1.index("\n")]
                            i,p=sock.getpeername()
                            if data == "bye":
                                    msg="\r"+record[(i,p)]+" left the
                                    conversation \33[0m\n"
                                    send_to_all(sock,msg)
                                    print ("Client (%s, %s) is offline" %
                                    (i,p)," [",record[(i,p)],"]")
                                    del record[(i,p)]
                                    connected_list.remove(sock)
                                    sock.close()
                                    continue
                            else:
                                    msg="\r"+record[(i,p)]+": " +data+"\n"
                                    send_to_all(sock,msg)
```

```
            except:
                    (i,p)=sock.getpeername()
                    send_to_all(sock, "\r\33[31m\33[1m"+record[(i,p)]+" left the
            conversation unexpectedly\33[0m\n")
                    print ("Client (%s, %s) is offline (error)" %(i,p)," [",record[(i,p)],"]\n")
                    del record[(i,p)]
                    connected_list.remove(sock)
                    sock.close()
    continue
server_socket.close()
```

Client:

```
import socket, select, string, sys
def display() :
you=" You: "
sys.stdout.write(you)
sys.stdout.flush()
def main():
 if len(sys.argv)<2:
 host = raw_input("Enter host ip address: ")
 else:
 host = sys.argv[1]
 port = 5001

 print("\t\t\t\t..... Client Working .....\n\n")
 name=raw_input(" Enter username: ")
 print("\n")
 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
 s.settimeout(2)



 try :
 s.connect((host, port))
 except :
 print (" Can't connect to the server ")
 sys.exit()

 s.send(name)
 display()
 while 1:
 socket_list = [sys.stdin, s]
```

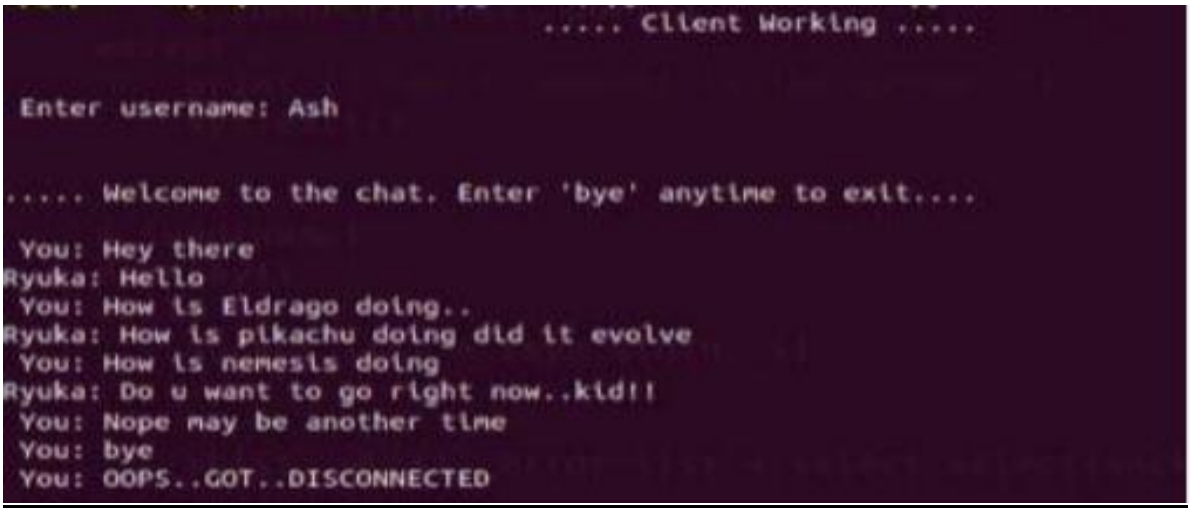
```
rList, wList, error_list = select.select(socket_list , [], [])

for sock in rList:

if sock == s:
data = sock.recv(4096)
if not data :
print ('OOPS..GOT..DISCONNECTED\n')
sys.exit()
else :
sys.stdout.write(data)
display()


else :
msg=sys.stdin.readline()
s.send(msg)
display()
if _name_ == "_main_":
main()
```

OUTPUT

Server

Client (RYUKA)

Client (ASH)



RESULT:

Successfully understood and executed a program to Implement Multi-user chat server using TCP as transport layer protocol.