NUST COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

Computer Networks

Project-Report

Submitted By

- M.Huzaifa Salik(289719)
- Fahad Abdullah(286137)
- Junaid Hassan(284925)
- CE-41 SYN 'A'

Submitted To

- Sir Umer Farooq

Brief Summary

We have implemented multi party chat application on TCP protocol.

The clients can share messages with each other and at the same time they can exchange files with each other.

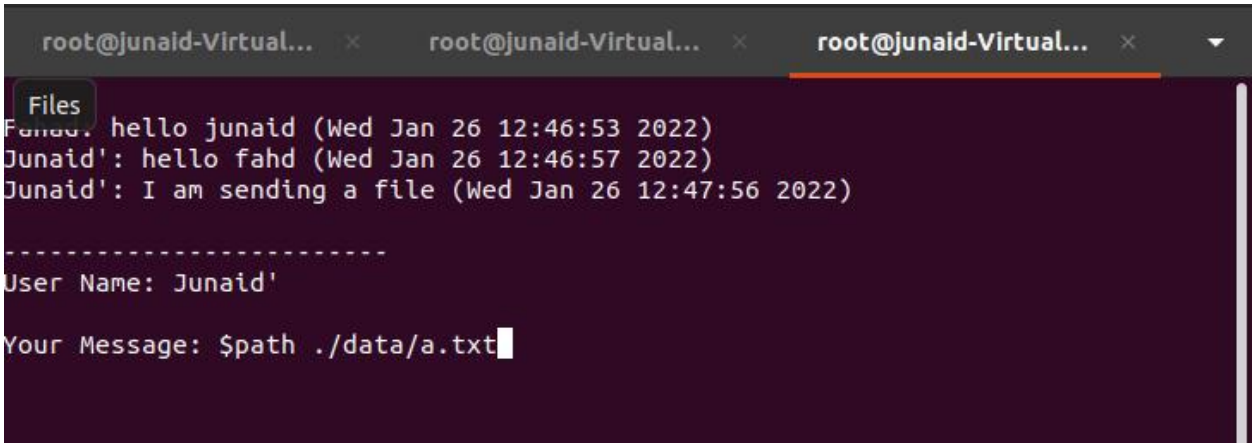
Working

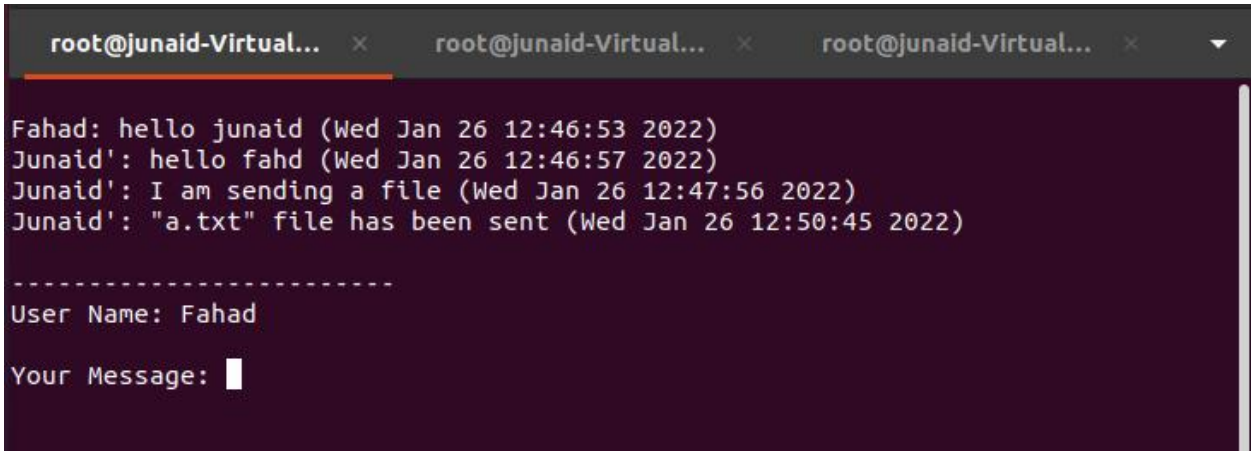
- We have used the threading method for providing every client its unique ID identifier.
- We have used the default port no 9002 for every client in order to connect with the TCP server.
- The message size we kept for sending is 500kb but it can be increased and can be sent otherwise.
- When a client sends a message it goes towards the server, those messages get save in an array in the server side, and then when other messages have been sent then the server continues to return those messages and then any client(connected with the server) can see those messages.
- In order to send the file by the client it has to type the particular command specified below.
- When the client sends a file then the server creates a folder server-data and stores that particular file in that folder, similarly a client-data folder has been created and when another client types the particular command to get that file then that file goes in that folder.
- Once the client sends a file then if the file transfer is successful/unsuccessful depending upon any sort of error the server returns the state.
- Once the client wants to exit from the group chat it can write the command listed below to exit from the chat and the remaining clients can continue their talking
- If we want to close down the server we can simply press ctrl+c to immediately close down the server

Commands

- **$path :** This command is used to send file, the client has to type this command and then also the path specified from which he has to send that particular file.

  Result





- **$refh :** This command can be used to refresh the chat e.g if one client sends a message in the chat then for showing it to the second client it has to type this command to get the latest chat data

  Result

```
root@junaid-Virtual...   root@junaid-Virtual...   root@junaid-Virtual...

Fahad: hello junaid (Wed Jan 26 12:46:53 2022)
Junaid': hello fahd (Wed Jan 26 12:46:57 2022)
Junaid': I am sending a file (Wed Jan 26 12:47:56 2022)
Junaid': "a.txt" file has been sent (Wed Jan 26 12:50:45 2022)

-------------------------
User Name: Junaid'

Your Message: $refh
```

```
root@junaid-Virtual...   root@junaid-Virtual...   root@junaid-Virtual...

Fahad: hello junaid (Wed Jan 26 12:46:53 2022)
Junaid': hello fahd (Wed Jan 26 12:46:57 2022)
Junaid': I am sending a file (Wed Jan 26 12:47:56 2022)
Junaid': "a.txt" file has been sent (Wed Jan 26 12:50:45 2022)
Fahad: okay' (Wed Jan 26 12:51:30 2022)

-------------------------
User Name: Junaid'

Your Message:
```

- **$getf :** This command is used to get the file from the server, basically when one client sends a file in the chat then that particular file is sent firstly towards the server and from there in order to access that file the other clients have to type this command and also type the name of the file.

  Result

```
root@junaid-Virtual...   root@junaid-Virtual...   root@junaid-Virtual...

Fahad: hello junaid (Wed Jan 26 12:46:53 2022)
Junaid': hello fahd (Wed Jan 26 12:46:57 2022)
Junaid': I am sending a file (Wed Jan 26 12:47:56 2022)
Junaid': "a.txt" file has been sent (Wed Jan 26 12:50:45 2022)
Fahad: okay' (Wed Jan 26 12:51:30 2022)

-------------------------
User Name: Fahad

Your Message: $getf a.txt
```

- **$exit :** This command is used to exit from the group chat once the client enters this command it can easily exit the group chat.

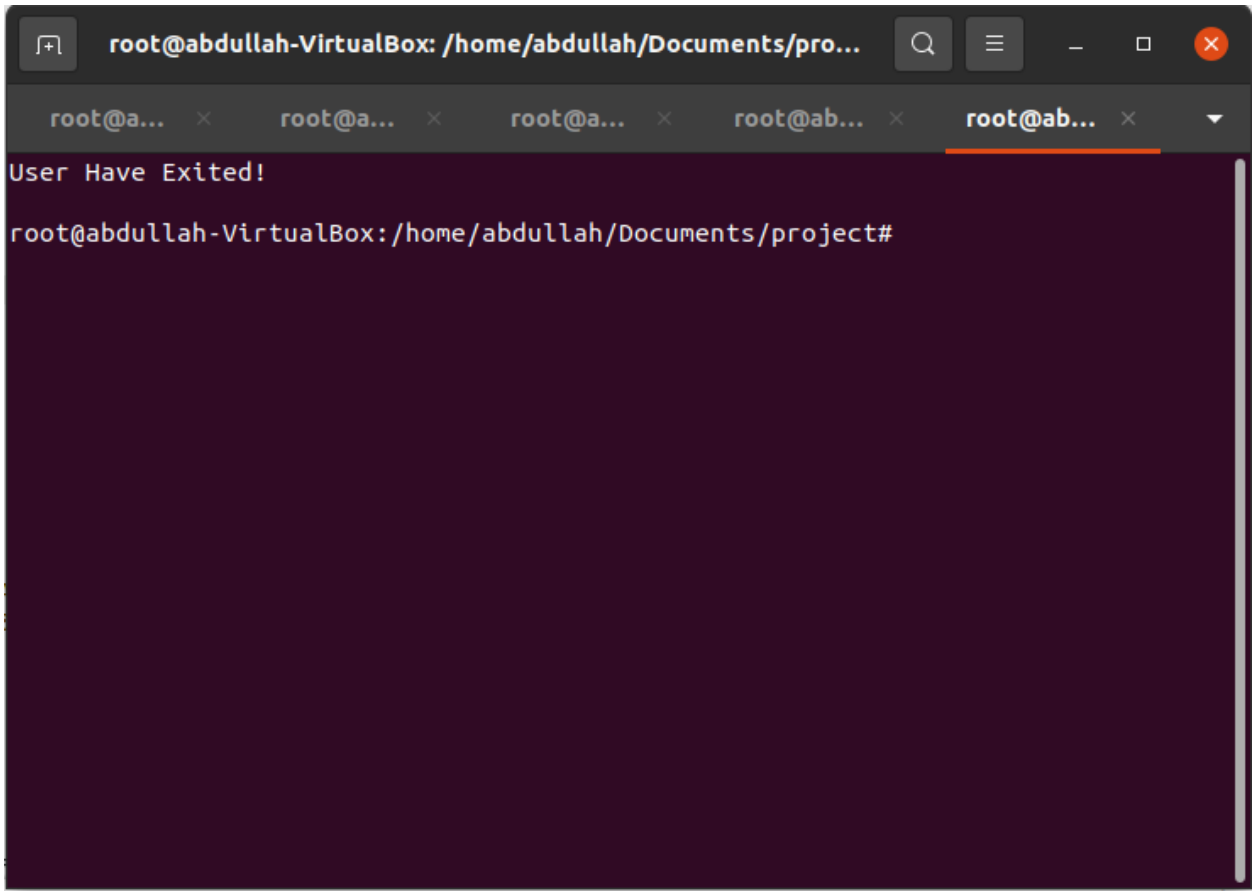
Result

```
root@abdullah-VirtualBox: /home/abdullah/Documents/pro...
root@a...   root@a...   root@a...   root@ab...   root@ab...

Fhaad: hello (Thu Jan 27 11:35:03 2022)
Huzaifa: hi (Thu Jan 27 11:35:11 2022)
Juandi: hi (Thu Jan 27 11:36:20 2022)
Juandi: "test.txt" file has been sent (Thu Jan 27 11:38:59 2022)
Huzaifa: "file.txt" file has been sent (Thu Jan 27 11:41:41 2022)
Ali: hi guys (Thu Jan 27 12:18:19 2022)

-------------------------
User Name: Ali

Your Message: $exit
```



Codes

Server

```
#include<iostream>

#include<cstring>

#include<stdio.h>

#include<stdlib.h>

#include<string>

#include <algorithm>


#include <chrono>
```

```
#include <ctime>



#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>

#include <pthread.h>

#include<fstream>
#include <sys/stat.h>

#define PORT 9002
#define TOTALCLIENTS 5
#define MAXMESSAGECHARARRAYLENGTH 512
#define MAXFILESIZE 512000 //500 KB

using namespace std;

//global variables
string fetchString="";

string getCurrentTime(){

	//get current time
	auto start = std::chrono::system_clock::now();
```

```
	// Some computation here
	auto end = std::chrono::system_clock::now();
	std::time_t end_time = std::chrono::system_clock::to_time_t(end);

	return string(std::ctime(&end_time)).erase(string(std::ctime(&end_time)).length()-1);
}

string getFilePath(string filePathString){
	//string filePath="$path ./send/q.txt";

	reverse(filePathString.begin(), filePathString.end());

	string pathOfFile="";
	for(int i=0;i<filePathString.length();i++){
		if(filePathString[i]!=':'){
			pathOfFile+=filePathString[i];
		}else{
			break;
		}
	}

	reverse(pathOfFile.begin(), pathOfFile.end());



	return pathOfFile;

}
```

```
string getFileNameFromPath(string filePath){
	//string filePath="$path ./send/q.txt";

	reverse(filePath.begin(), filePath.end());

	string nameOfFile="";
	for(int i=0;i<filePath.length();i++){
		if(filePath[i]!='/'){
			nameOfFile+=filePath[i];
		}else{
			break;
		}
	}

	reverse(nameOfFile.begin(), nameOfFile.end());



	return nameOfFile;

}

string getUserNameFromPath(string recvString){

	string nameOfUser="";
	for(int i=0;i<recvString.length();i++){
		if(recvString[i]!=':'){
```

```
                        nameOfUser+=recvString[i];
                }else{
                        break;
                }
        }

        return nameOfUser;

}

void messagesReciever(int client_socket_id){

while(1){

        //system(cls) do not work on linux
        //system("clear");

        //5- data exchanage
        char clientSendMsg[MAXMESSAGECHARARRAYLENGTH];
        recv(client_socket_id,&clientSendMsg,sizeof(clientSendMsg),0);




        //message starting with $ is a command
        if(string(clientSendMsg)!="$refh"&&clientSendMsg[0]!='$'&&getFilePath(string(clientSe
ndMsg)).substr(0, 5) != "$path"){
```

```
fetchString=fetchString+"\n"+string(clientSendMsg);

cout<<endl<<fetchString<<endl;

}else if(getFilePath(string(clientSendMsg)).substr(0, 5) == "$path"){
fetchString=fetchString+"\n"+getUserNameFromPath(string(clientSendMsg))+": \""+getFileNameFromPath(string(clientSendMsg))+"\" file has been sent ("+getCurrentTime()+")";
}else{
//do nothing and code will automatically refresh chat
}


//recv the file here


if(getFilePath(string(clientSendMsg)).substr(0, 5) == "$path"){
fstream fileRecv;

fileRecv.open("./server_data/"+getFileNameFromPath(string(clientSendMsg)), ios::out | ios::trunc | ios::binary);

if(fileRecv.is_open()){
  //cout<<"[LOG] : File is Opened (Recv)";

//1024*500 = 500 kb
        char buffer[MAXFILESIZE] = {};
int valread = read(client_socket_id , buffer, MAXFILESIZE);
```

```
        //cout<<"Data received = "<<valread<<" bytes"<<endl;
        //cout<<"Saving data to file."<<endl;

        fileRecv<<buffer;
        //cout<<"FILE SAVE SUCCESSFULL"<<endl;

       }
       else{
          cout<<"[ERROR] : File loading failed (Recv)";
          exit(EXIT_FAILURE);
       }

     fileRecv.close();

        }else if(string(clientSendMsg).substr(0, 5) == "$getf"){

                fstream fileSend;

        fileSend.open("./server_data/"+string(clientSendMsg).substr(6,string(clientSendMsg).le
ngth()), ios::in | ios::binary);
       if(fileSend.is_open()){
         //cout<<"[LOG] : File is ready to Transmit";

        std::string contents((std::istreambuf_iterator<char>(fileSend)),
std::istreambuf_iterator<char>());
        //cout<<"Size of data to be transmitted = "<<contents.length()<<" Bytes."<<endl;

        //cout<<"Sending Data..."<<endl;
```

```
    int bytes_sent = send(client_socket_id , contents.c_str() , contents.length() , 0 );
    //cout<<"Size of data transmitted = "<<bytes_sent<<" Bytes."<<endl;

    //cout<<"FILE TRANSFER SUCCESSFULL"<<endl;

   }
   else{
      cout<<"[ERROR] : File loading failed, Exititng";
      exit(EXIT_FAILURE);
   }

fileSend.close();

    }else if(string(clientSendMsg).substr(0, 5) == "$exit"){

                close(client_socket_id);

    }else{
                //do nothing
    }

    if(string(clientSendMsg).substr(0, 5) != "$getf"){
    //convert string message to char array
    int n = fetchString.length();
    char char_array[n + 1];
    strcpy(char_array, fetchString.c_str());
```

```
        //send grp chat to client that typed message
        send(client_socket_id,char_array,sizeof(char_array),0);
        }

}

}

void* recieverThreadFunction(void *mSocket_id) {



        long socket_id=(long) mSocket_id;

        messagesReciever(socket_id);

        return 0;



}


//THIS IS SERVER
int main(){

        //starting code
```

```
	system("clear");
	//create data folder
	mkdir("./server_data/", 0777);

	cout<<"Server is Running"<<endl<<endl;

	//1- create socket
	int server_socket_id=socket(AF_INET,SOCK_STREAM,0);

	//AF_INET: IPv4
	//SOCK_STREAM: TCP

	if(server_socket_id==-1){
		cout<<"ERROR WHILE CREATING SCOKET"<<endl;
	}

	//this prevents error binding for rerun
	int enable = 1;
	if (setsockopt(server_socket_id, SOL_SOCKET, SO_REUSEADDR, &enable, sizeof(enable))
< 0){
		cout<<("setsockopt(SO_REUSEADDR) failed");
	}

	//2- bindind socket
	//bind socket to a port of address defined by this structure
	struct sockaddr_in addr;
```

```
	addr.sin_addr.s_addr = INADDR_ANY;
	addr.sin_family = AF_INET; //IPv4
	addr.sin_port = htons(PORT); //Bind to port 9002

	if (bind(server_socket_id, (struct sockaddr *) &addr, sizeof(addr)) == -1) {
		cout<<"ERROR WHILE BINDING SCOKET"<<endl;
	}

	//3- listening
	//TOTALCLIENTS //max no of clients

	if (listen(server_socket_id, TOTALCLIENTS) == -1) {
		cout<<"ERROR WHILE LISTENING"<<endl;

	}

	/*int yes=1;

	if (setsockopt(server_socket_id, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes)) == -1)
{
		cout<<"setsockopt";
		exit(1);
	}*/

	//4- accept
	long client_socket_id[TOTALCLIENTS];
```

```
//For changing TOTALCLIENTS must add approperate number of object here
pthread_t tid0;
pthread_t tid1;
pthread_t tid2;
pthread_t tid3;
pthread_t tid4;
pthread_t * pthreads[] = {&tid0,&tid1,&tid2,&tid3,&tid4};
//pthread_t * pthreads[TOTALCLIENTS];

for (int i = 0; i < TOTALCLIENTS; i++){

struct sockaddr_in cliaddr;
socklen_t cliaddr_len = sizeof(cliaddr);

client_socket_id[i] = accept(server_socket_id, (struct sockaddr *) &cliaddr,
&cliaddr_len);
if (client_socket_id[i] == -1) {
// an error occurred

}

pthread_create(pthreads[i],NULL,recieverThreadFunction,(void *) client_socket_id[i]);
}

/*char clientSendMsg2[MAXMESSAGECHARARRAYLENGTH]="Hello Client!";
```

```
        send(client_socket_id,clientSendMsg2,sizeof(clientSendMsg2),0);*/

        close(server_socket_id);

        for (int i = 0; i < TOTALCLIENTS; i++){
        close(client_socket_id[i]);
        }

        return 0;
}
```

Client

```
#include<iostream>
#include<cstring>
#include<stdio.h>
#include<stdlib.h>
#include<string>
#include <algorithm>

#include <chrono>
#include <ctime>



#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
```

```
#include<fstream>
#include <sys/stat.h>

#define PORT 9002
#define MAXFETCHCHARARRAYLENGTH 4096
#define MAXFILESIZE 512000 //500 KB



using namespace std;

//global variables
string chatString="";
bool isRunning=true;
bool isFirstTimeRunning=true;

string getFilePath(string filePathString){
	//string filePath="$path ./send/q.txt";

	reverse(filePathString.begin(), filePathString.end());

	string pathOfFile="";
	for(int i=0;i<filePathString.length();i++){
		if(filePathString[i]!=':'){
			pathOfFile+=filePathString[i];
		}else{
			break;
		}
```

```
        }

        reverse(pathOfFile.begin(), pathOfFile.end());


        return pathOfFile;

}

string getClientName(){

        string userName;
        cout<<"Enter Your Name: ";
        getline(cin, userName);

        if (userName.length()>0)
        {
                userName[0] = std::toupper(userName[0]) ;
                for (size_t i = 1; i < userName.length(); i++){
                userName[i] = std::tolower(userName[i]);
                }
        }

        return userName;
}

string getCurrentTime(){
```

```
        //get current time
        auto start = std::chrono::system_clock::now();
        // Some computation here
        auto end = std::chrono::system_clock::now();
        std::time_t end_time = std::chrono::system_clock::to_time_t(end);

        return string(std::ctime(&end_time)).erase(string(std::ctime(&end_time)).length()-1);
}

void messagesSender(int server_socket_id,string userName){

while(1){

        cout<<"-------------------------"<<endl;
        cout<<"User Name: "<<userName<<endl<<endl;

        //get message

        string message="";
        if(isRunning){

        if(!isFirstTimeRunning){
        cout<<"Your Message: ";
        getline(cin, message);
        }else{
        message="$refh";
```

```
isFirstTimeRunning=false;
}

system("clear");
}else{

system("clear");
cout<<"User Have Exited!"<<endl<<endl;
//break;

}



if(message[0]!='$'){
//concatinate message with user name
message=userName+": "+message+" ("+getCurrentTime()+") ";
}else if(message.substr(0, 5) == "$path"){
message=userName+":"+message;
}else{
//this is a command and send it as it is to the server
//commands are handled here
}




//convert string message to char array
int n = message.length();
```

```
char char_array[n + 1];
strcpy(char_array, message.c_str());



//send message
//3- data exchanage
send(server_socket_id,char_array,sizeof(char_array),0);

//send the file here



if(getFilePath(message).substr(0, 5) == "$path"){




fstream fileSend;

fileSend.open(getFilePath(message).substr(6,message.length()), ios::in | ios::binary);
if(fileSend.is_open()){
  //cout<<"[LOG] : File is ready to Transmit";

std::string contents((std::istreambuf_iterator<char>(fileSend)),
std::istreambuf_iterator<char>());
//cout<<"Size of data to be transmitted = "<<contents.length()<<" Bytes."<<endl;

//cout<<"Sending Data..."<<endl;
```

```
int bytes_sent = send(server_socket_id , contents.c_str() , contents.length() , 0 );
//cout<<"Size of data transmitted = "<<bytes_sent<<" Bytes."<<endl;

//cout<<"FILE TRANSFER SUCCESSFULL"<<endl;

}
else{
   cout<<"[ERROR] : File loading failed, Exititng";
   exit(EXIT_FAILURE);
}

fileSend.close();

}else if(message.substr(0, 5) == "$getf"){

fstream fileRecv;

fileRecv.open("./client_data/"+message.substr(6,message.length()), ios::out | ios::trunc
| ios::binary);

if(fileRecv.is_open()){
  //cout<<"[LOG] : File is Opened (Recv)";

//1024*500 = 500 kb
        char buffer[MAXFILESIZE] = {};
int valread = read(server_socket_id , buffer, MAXFILESIZE);
//cout<<"Data received = "<<valread<<" bytes"<<endl;
```

```
    //cout<<"Saving data to file."<<endl;

    fileRecv<<buffer;
    //cout<<"FILE SAVE SUCCESSFULL"<<endl;

   }
   else{
      cout<<"[ERROR] : File loading failed (Recv)";
      exit(EXIT_FAILURE);
   }

fileRecv.close();

    }else{
              //do nothing
    }


    if(message.substr(0, 5) != "$getf"){
    //get back whole group chat
    char fetchCharArray[MAXFETCHCHARARRAYLENGTH];
    recv(server_socket_id,&fetchCharArray,sizeof(fetchCharArray),0);
    chatString=string(fetchCharArray);
    }

    cout<<chatString<<endl<<endl;
```

```
        if(message.substr(0, 5) == "$exit"){

                system("clear");
                isRunning=false;
                message="";

        }




}

}


//THIS IS CLIENT
int main(){

        //starting code
        system("clear");
        //create data folder
        mkdir("./client_data/", 0777);

        cout<<"Client is Running"<<endl<<endl;

        //1- create socket
        int server_socket_id=socket(AF_INET,SOCK_STREAM,0);
```

```
//AF_INET: IPv4
//SOCK_STREAM: TCP

if(server_socket_id==-1){
	cout<<"ERROR WHILE CREATING SCOKET"<<endl;
}

//2- connect
struct sockaddr_in s_addr;
s_addr.sin_family = AF_INET; //Protocol family
s_addr.sin_port = htons(PORT); //Remember the port number in server
//application!
//inet_aton("127.0.0.1", &s_addr.sin_addr); =>Not Working
// INADDR_ANY = Server address on local machine
s_addr.sin_addr.s_addr = INADDR_ANY;

if (connect(server_socket_id, (struct sockaddr *) &s_addr, sizeof(s_addr)) == -1)
{
	cout<<"ERROR WHILE CONNECTING"<<endl;
}

string userName=getClientName();
cout<<endl;

system("clear");

messagesSender(server_socket_id,userName);
```

```
        /*char clientSendMsg2[256];
        recv(socket_id,&clientSendMsg2,sizeof(clientSendMsg2),0);
        cout<<clientSendMsg2<<endl;*/

        close(server_socket_id);



        return 0;
}
```