

A COURSE PROJECT REPORT

By

DEVASHISH KRISHNA (RA2011031010062)
RIYA SINGH (RA2011031010044)
ADITI MISHRA(RA2011031010041)
VANI KUMAR (RA2011031010065)

Under the guidance of

Dr. S. Thenmalar

In partial fulfilment for the Course

of

18CSC302J - COMPUTER NETWORKS

in Computer Science Engineering Specialization in IT



FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chenpalattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "**Multimedia Messaging Application**" is the bonafide work of **DEVASHISH KRISHNA (RA2011031010062)**, **RIYA SINGH (RA2011031010044)**, **ADITI MISHRA (RA2011031010041)**, **VANI KUMAR (RA2011031010065)** who carried out the project work under my supervision.

Signature

Ms.S.Thenmalar
Associate Professor
NWC

SRM Institute of Science and Technology

ABSTRACT

We proposed to build an Multimedia Application. Teleconferencing or Chatting, is a method of using technology to bring people and ideas "together" despite of the geographical barriers. The technology has been available for years but the acceptance it was quite recent, Our project is an example of a chat server. It is made up of 2 applications the client application, which runs on the user's Pc and server application, which runs on any Pc on the network. To start chatting client should get connected to server where they can practice two kinds of chatting, public one (message is broadcasted to all connected users) and private one (between any 2 users only) and during the last one security measures were taken

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal**, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Ms.S.Thenmalar, professor , NWC**, for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

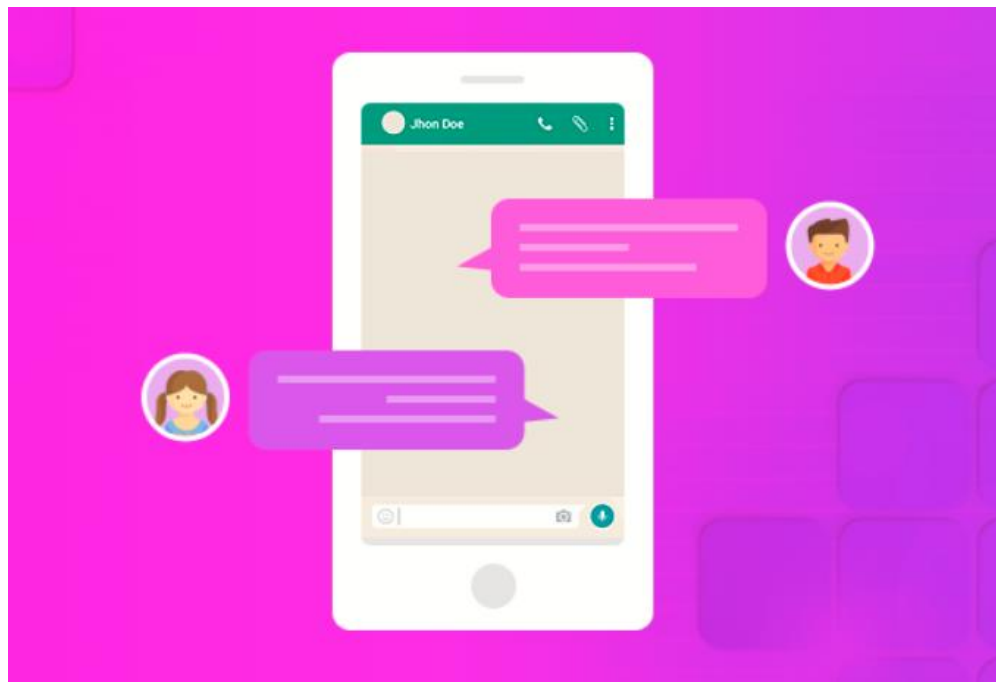
TABLE OF CONTENTS

CHAPTERS	CONTENTS
1.	ABSTRACT
2.	INTRODUCTION
3.	REQUIREMENT ANALYSIS
4.	IMPLEMENTATION
5.	EXPERIMENT RESULTS & ANALYSIS
	5.1. RESULTS
	5.2. RESULT ANALYSIS
6.	CONCLUSION & FUTURE ENHANCEMENT
7.	REFERENCES

1. INTRODUCTION

1.1 Aim of the project

To Implement Secure multimedia chat application using FTP and full Duplex for chatting, sending photos and text file with friends by using C language.



1.2 Multimedia Chat Application

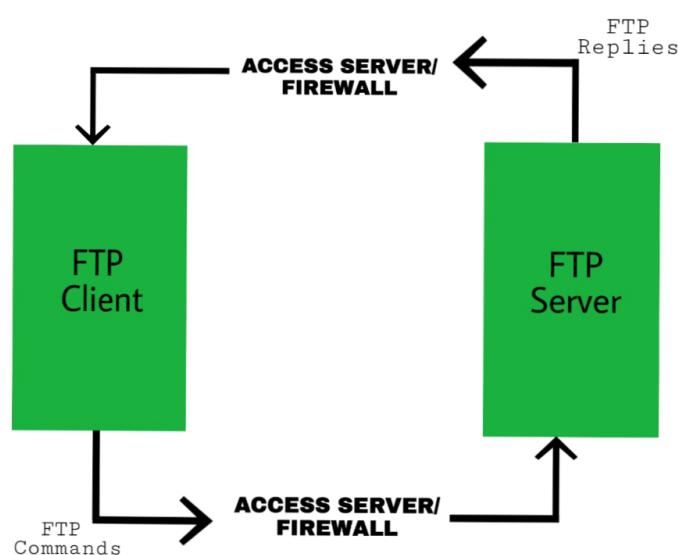
Instant messaging applications can store messages with either local-based device storage (e.g WhatsApp, Viber, Line, WeChat, Signal etc.) or cloud-based server storage (e.g Telegram, Skype, Facebook Messenger, Google Hangouts, Discord, Slack etc.).

In view of, a modern chat app must possess noteworthy functionalities which can be adaptable to any type of chat solutions. In recent times, it has been found that creating chat app by using C language. An ideal chat app has the potential of offering solutions to send files, talk without interruption and can also send photos by using C language.

The FTP connection is established between two systems and they communicate with each other using a network. So, for the connection, the user can get permission by providing the credentials to the FTP server or can use anonymous FTP.

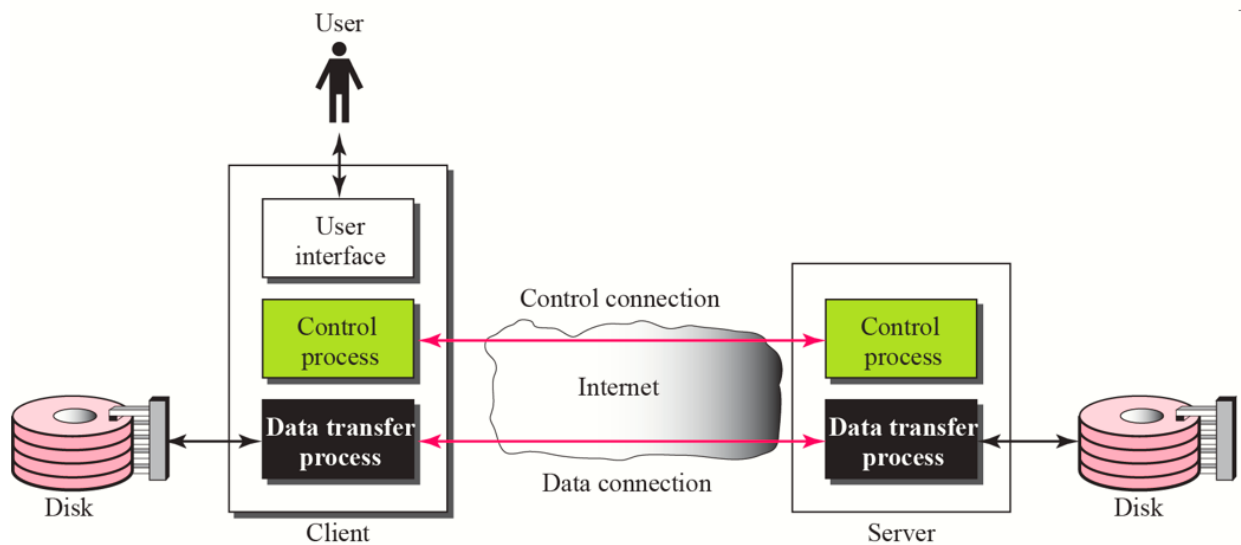
When an FTP connection is established, there are two types of communication channels are also established and they are known as command channel and data channel. The command channel is used to transfer the commands and responses from client to server and server to client. FTP uses the same approach as TELNET or SMTP to communicate across the control connection. It uses the NVT ASCII character set for communication. It uses port number 21. Whereas the data channel is used to actually transfer the data between client and server. It uses port number 20.

The FTP client using the URL gives the FTP command along with the FTP server address. As soon as the server and the client get connected to the network, the user logs in using User ID and password. If the user is not registered with the server, then also he/she can access the files by using the anonymous login where the password is the client's email address. The server verifies the user login and allows the client to access the files. The client transfers the desired files and exits the connection. The figure below shows the working of FTP.



URL: <https://www.geeksforgeeks.org/file-transfer-protocol-ftp/amp/>

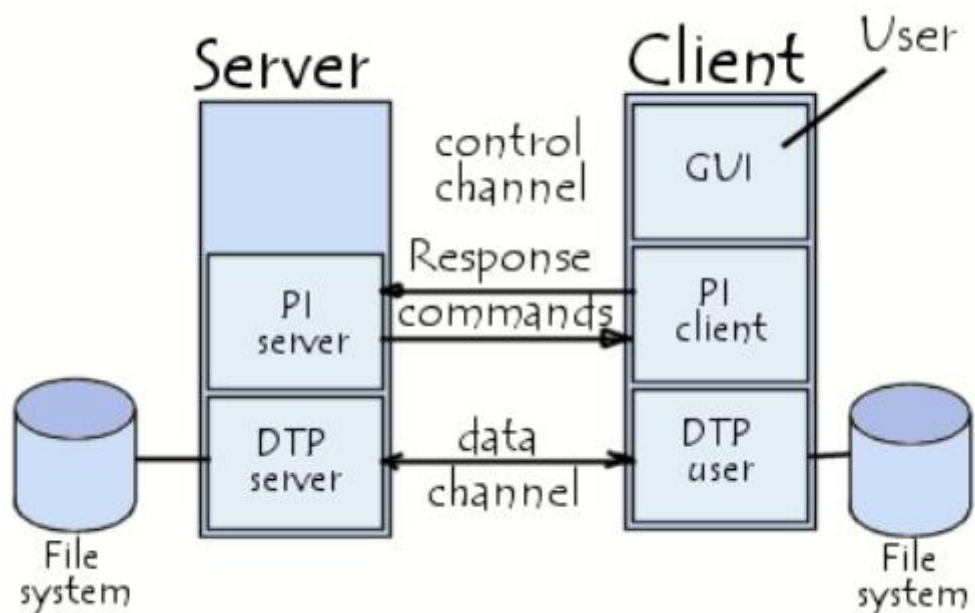
In the below figure you can see the basic functions done by Client-Server programs using FTP protocol for establishment of a successful connection, communication and finally connection teardown or termination.



In this configuration, the protocol imposes that the control channels remain open throughout the data transfer. So a server can stop transmission if the control channel is broken during transmission. During an FTP connection, two transmission channels are open:

A channel for commands (control channel).

A channel for data.



2.REQUIREMENTS

3.1 HMI principles implemented in the project

1. Place Users at the Centre

As always, the first UI design principle is to focus on people (or, the “user” as we all say). A good user interface is easy and natural to use, avoids confusing the user, and does what the user needs. Learning about human-centered design will help you achieve the right mindset for the best interfaces and focus on people first, and design second.

2. Strive for Clarity

- The purpose of the user interface is to allow the user to interact with the website or application (or, more generally in broader design, any product). Avoid anything that confuses people or doesn't help them interact.

3. Minimise Actions and Steps Per Screen

- Streamline tasks and actions so they can be done in as few steps as possible. Each screen should have one primary focus. Keep the primary action front and centre and move secondary actions deeper on a page or give them lighter visual weight and the right typography.

4. Aim for Simplicity

- Classics exist for a reason; they're time less and never go out of style, though they do benefit from modern touches. A user interface should be simple and elegant.

5. Be Consistent

- Consistency creates familiarity, and familiar interfaces are naturally more usable. Consistent design reduces friction for the user. A consistent design is predictable. Predictable design means it's easy to understand how to use functions without instruction.

6. Your Goal : Make Your User Interface Design Invisible

- Don't draw attention to your user interface. A great UI allows people to use the product without friction, not spend time figuring out how to interact with the product.

7. Provide Useful Feedback

- Feedback can be visual, audio (the ding of a new message alert), or sense of touch. Every action should have feedback to indicate whether the action was successful or not.
- Feedback helps to answer questions in four areas:
 - Location: You are here.
 - Status: What's going on? Is it still going on?
 - Future status: What's next?
 - Outcomes & Results: Hey, what happened?
- Hovering over an avigation item that then changes colour indicates an item is clickable. Buttons should look like buttons. Feedback lets the user know If they're doing the right thing(or the wrong thing).

8. Reduce Cognitive Load

- Many of these UI design principles serve to reduce cognitive load for users. Basically, don't make users think(also a useful UX design principle as well).

9. Make It Accessible

- UI designs need to take into account accessibility issues. Online, this often means ensuring the visibly impaired can access and use the product.

10. Flexible

- Create a UI that will work and look great across multiple platforms. It may have to bet weaked depending on the form factor of adeviceandits

Description

Socket programming

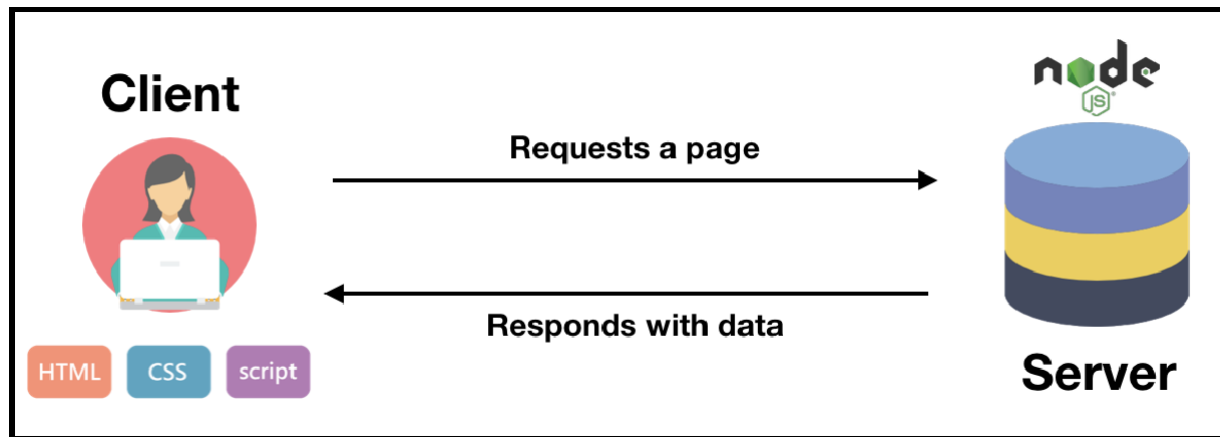
Sockets can be thought of as endpoints in a communication channel that is bi-directional and establishes communication between a server and one or more clients. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates 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.

FTP Server

The primary purpose of an FTP server is to allow users to upload and download files. An FTP server is a computer that has a file transfer protocol (FTP) address and is dedicated to receiving an FTP connection. FTP is a protocol used to transfer files via the internet between a server (sender) and a client (receiver). An FTP server is a computer that offers files available for download via an FTP protocol, and it is a common solution used to facilitate remote data sharing between computers.

Duplex Communication

Full-duplex data transmission means that data can be transmitted in both directions on a signal carrier at the same time. For example, on a local area network with a technology that has full-duplex transmission, one workstation can be sending data on the line while another workstation is receiving data. Full-duplex transmission implies a bidirectional line that can move data in both directions simultaneously.



Objectives

Objectives of Multimedia chat project are given below :

- Create a Multimedia Chat using FTP protocol to chat with friends, send photos and also sends text files.
- By creating chat application using C program gives good hand on it.
- Gives a great knowledge in Computer Networking by using FTP protocol.

2. IMPLEMENTATION

Chat Server

/* Implementing C language code for Full Duplex */

```
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>

int main(int argc,char *argv[])
{
int clientSocketDescriptor,socketDescriptor;

struct sockaddr_inserverAddress,clientAddress;
socklen_tclientLength;

char recvBuffer[1000],sendBuffer[1000];
pid_tcpid;
bzero(&serverAddress,sizeof(serverAddress));

serverAddress.sin_family=AF_INET;
serverAddress.sin_addr.s_addr=htonl(INADDR_ANY);
serverAddress.sin_port=htons(5500);

socketDescriptor=socket(AF_INET,SOCK_STREAM,0);

bind(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));

listen(socketDescriptor,5);
printf("%s\n","Server is running ...");

clientSocketDescriptor=accept(socketDescriptor,(struct sockaddr*)&clientAddress,&clientLength);

cpid=fork();

if(cpid==0)
{
while(1)
{
bzero(&recvBuffer,sizeof(recvBuffer));
```

```

recv(clientSocketDescriptor,recvBuffer,sizeof(recvBuffer),0);
printf("\nCLIENT : %s\n",recvBuffer);
}
}
else
{
while(1)
{

bzero(&sendBuffer,sizeof(sendBuffer));
printf("\nType a message here ... ");

fgets(sendBuffer,10000,stdin);

send(clientSocketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
printf("\nMessage sent !\n");
}
}
return 0;
}

/* Successfully implemented multimedia chat application by using FTP */

#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5033
#define MAX 60
int i, j, tem;
char buff[4096], t;
FILE *f1;
int main(int afg, char *argv)
{
    int sockfd, newsockfd, clength;
    struct sockaddr_inserv_addr,cli_addr;
    char t[MAX], str[MAX];
    strcpy(t,"exit");
    sockfd=socket(AF_INET, SOCK_STREAM,0);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(SERV_TCP_PORT);
    printf("\nBinded");
    bind(sockfd,(struct sockaddr*)&serv_addr, sizeof(serv_addr));
    printf("\nListening...");
    listen(sockfd, 5);

```

```

length=sizeof(cli_addr);
newsockfd=accept(sockfd,(struct sockaddr*) &cli_addr,&length);
close(sockfd);
read(newsockfd, &str, MAX);
printf("\nClient message\n File Name : %s\n", str);
f1=fopen(str, "r");
while(fgets(buff, 4096, f1)!=NULL)
{
    write(newsockfd, buff,MAX);
}
printf("\n");
fclose(f1);
printf("\nFile Transferred\n");
return 0;
}

```

Chat Client

```

/* Implementing C language code for Full Duplex */
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include "netdb.h"
#include "arpa/inet.h"
int main()
{
    int socketDescriptor;
    struct sockaddr_in serverAddress;
    char sendBuffer[1000],recvBuffer[1000];
    pid_t cpid;
    bzero(&serverAddress,sizeof(serverAddress));
    serverAddress.sin_family=AF_INET;
    serverAddress.sin_addr.s_addr=inet_addr("127.0.0.1");
    serverAddress.sin_port=htons(5500);
    socketDescriptor=socket(AF_INET,SOCK_STREAM,0);
    connect(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));
    cpid=fork();
    if(cpid==0)
    {
        while(1)
        {
            bzero(&sendBuffer,sizeof(sendBuffer));
            printf("\nType a message here ... ");
            fgets(sendBuffer,10000,stdin);
            send(socketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
            printf("\nMessage sent !\n");
        }
    }
}

```

```

}
else
{
while(1)
{
bzero(&recvBuffer,sizeof(recvBuffer));

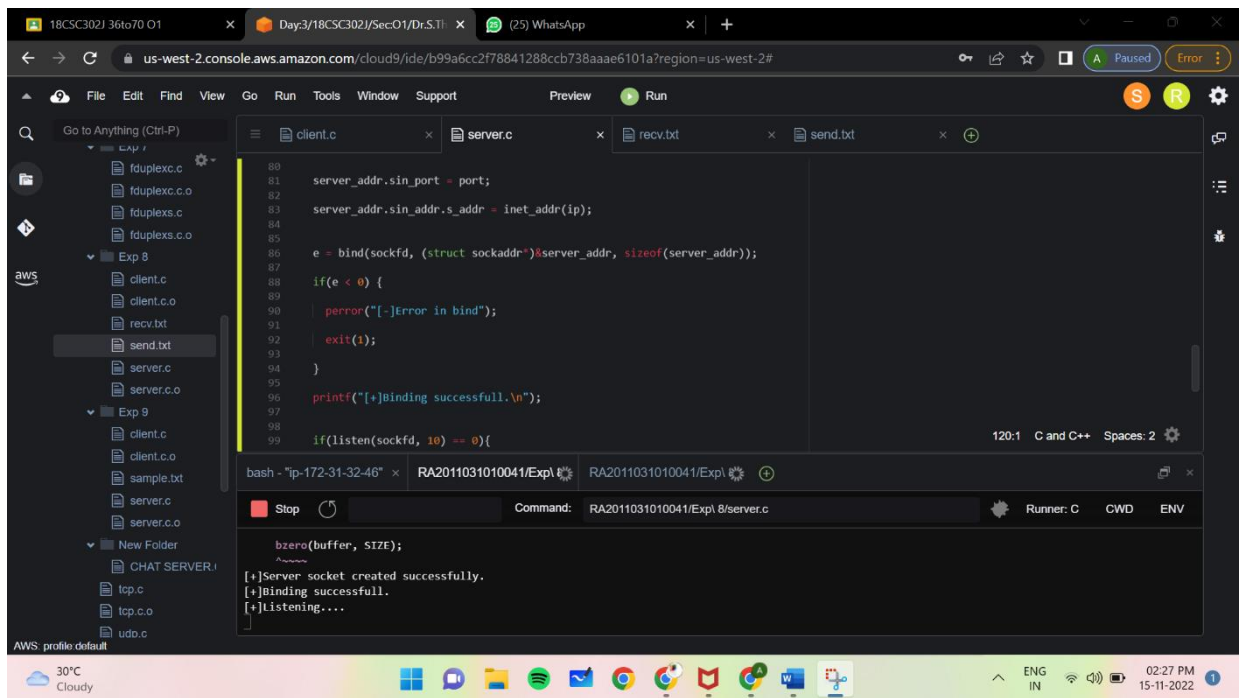
recv(socketDescriptor,recvBuffer,sizeof(recvBuffer),0);
printf("\nSERVER : %s\n",recvBuffer);
}
}
return 0;
}

/* Successfully implemented multimedia chat application by using FTP */

#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#define SERV_TCP_PORT 5033
#define MAX 60
int main(int arg,char*argv[])
{ int sockfd,n;
  struct sockaddr_in serv_addr;
  struct hostent*server;
  char send[MAX],recvline[MAX],s[MAX],name[MAX];
sockfd=socket(AF_INET,SOCK_STREAM,0);
serv_addr.sin_family=AF_INET;
serv_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
serv_addr.sin_port=htons(SERV_TCP_PORT);
  connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
printf("\nEnter the source file name : \n");
scanf("%s",send);
  write(sockfd,send,MAX);
  while((n=read(sockfd,recvline,MAX))!=0)
  { printf("%s",recvline); }
  close(sockfd);
  return 0;
}

```


Output:

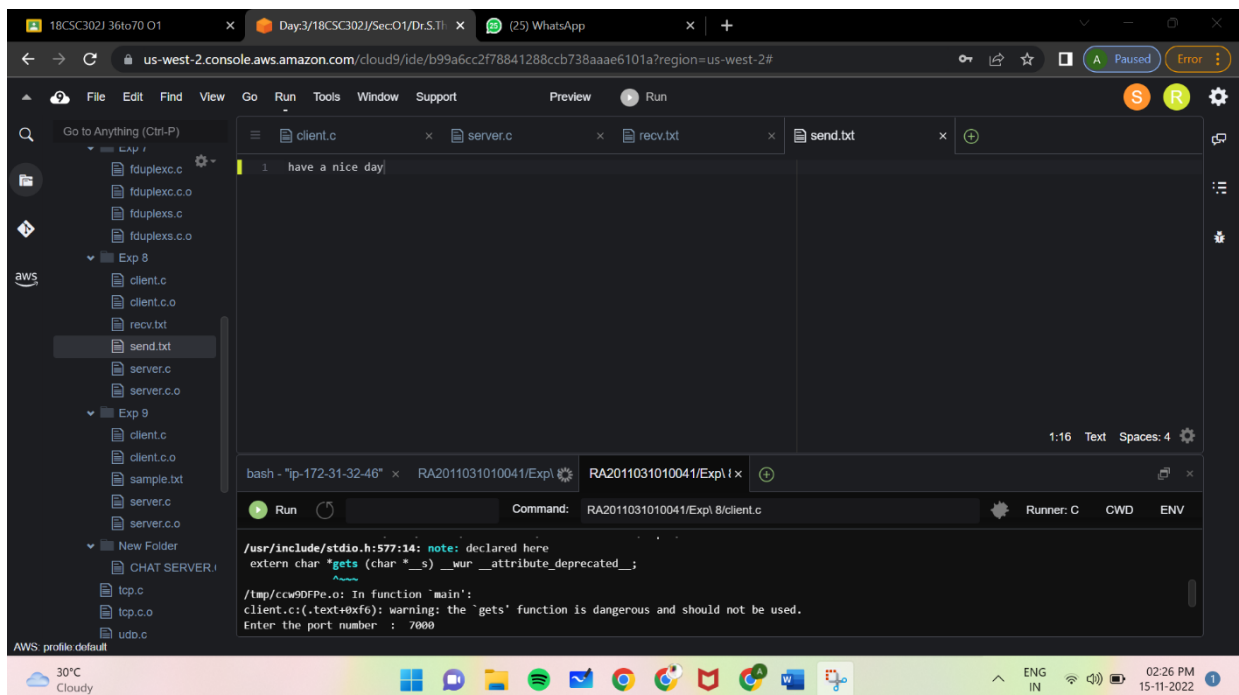


The screenshot shows the AWS Cloud9 IDE interface. The file explorer on the left lists files for 'Exp 8' and 'Exp 9'. The editor displays 'server.c' with the following code:

```
80 server_addr.sin_port = port;
81
82 server_addr.sin_addr.s_addr = inet_addr(ip);
83
84
85
86 e = bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
87
88 if(e < 0) {
89     perror("[-]Error in bind");
90     exit(1);
91 }
92
93 printf("[+]Binding successfull.\n");
94
95
96 if(listen(sockfd, 10) == 0){
97
98
99
```

The terminal at the bottom shows the command 'RA2011031010041/Exp1 8/server.c' and its output:

```
bash - "ip-172-31-32-46" x RA2011031010041/Exp1 8/server.c
bzero(buffer, SIZE);
[+]Server socket created successfully.
[+]Binding successfull.
[+]listening....
```

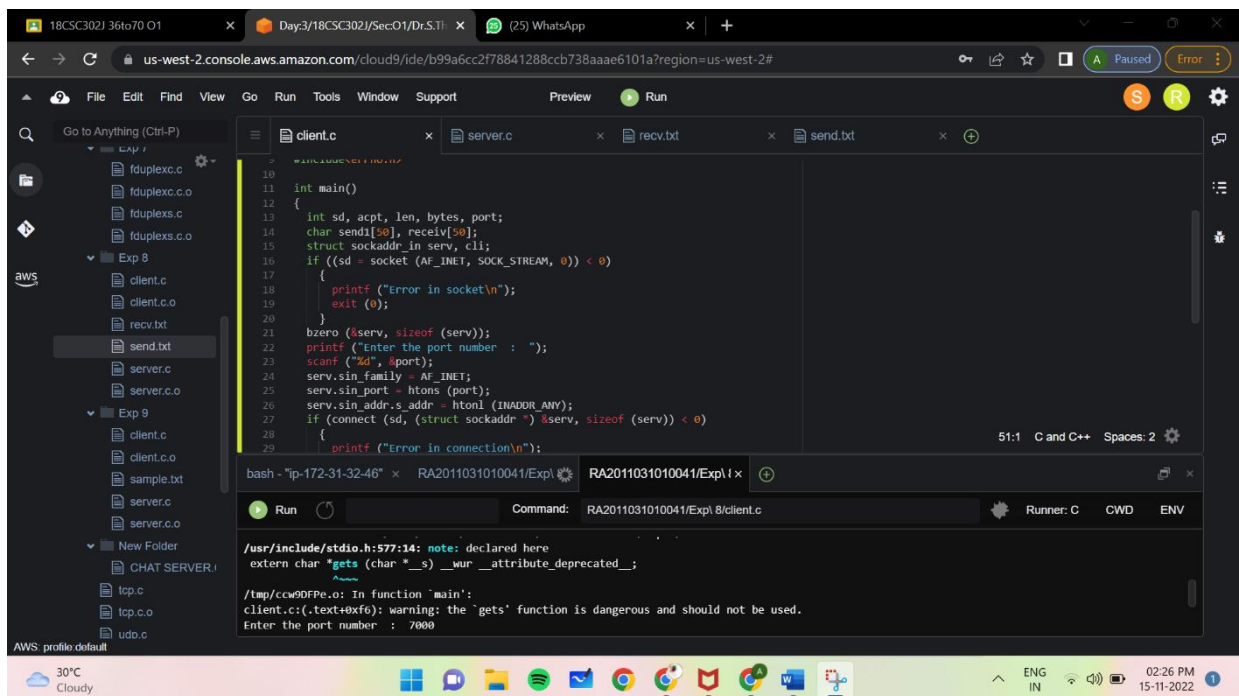
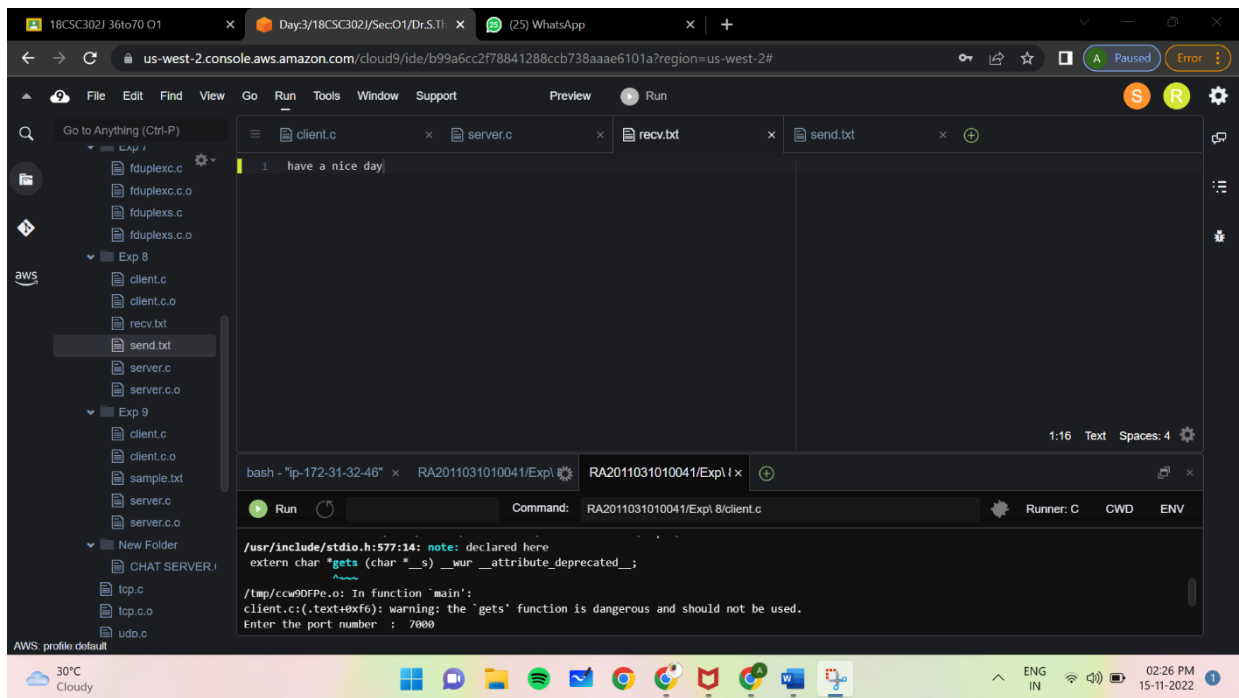


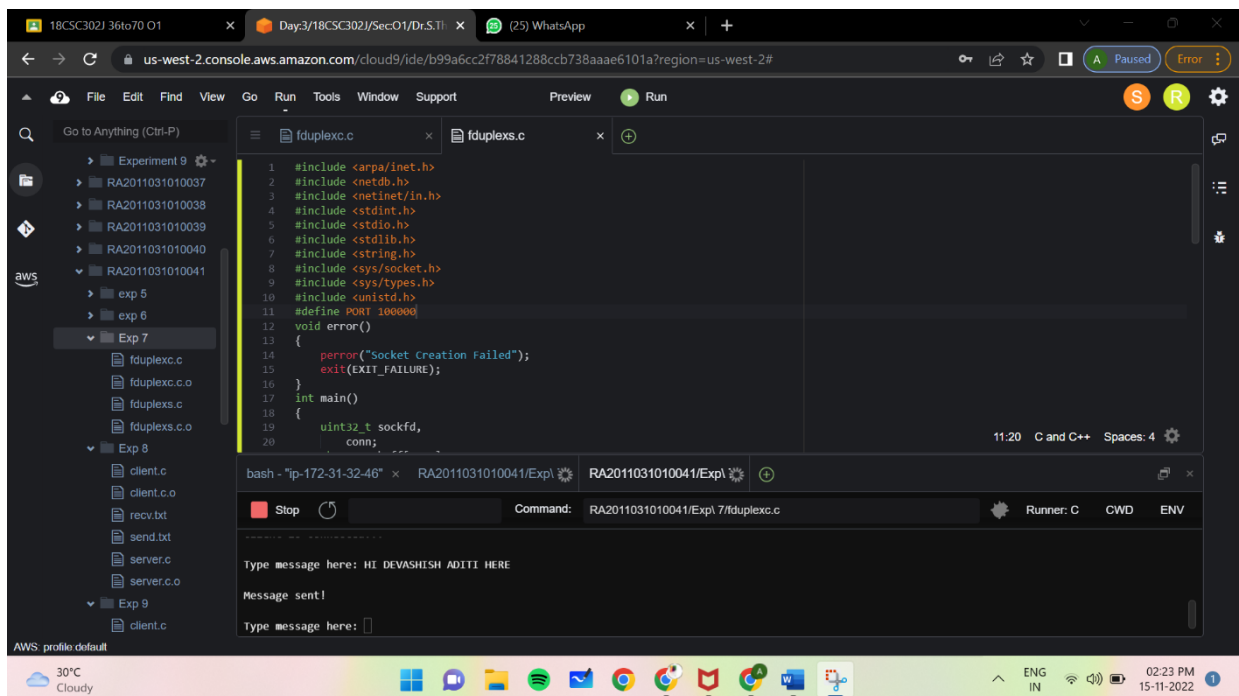
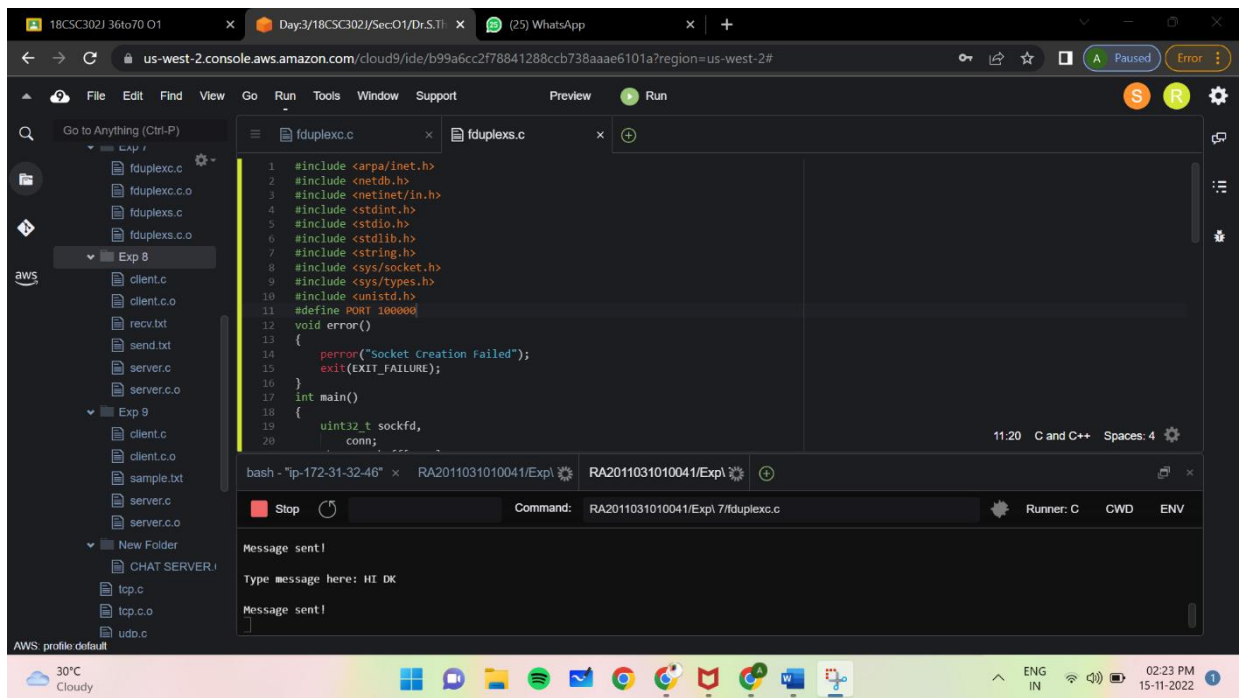
The screenshot shows the AWS Cloud9 IDE interface. The file explorer on the left lists files for 'Exp 8' and 'Exp 9'. The editor displays 'client.c' with the following code:

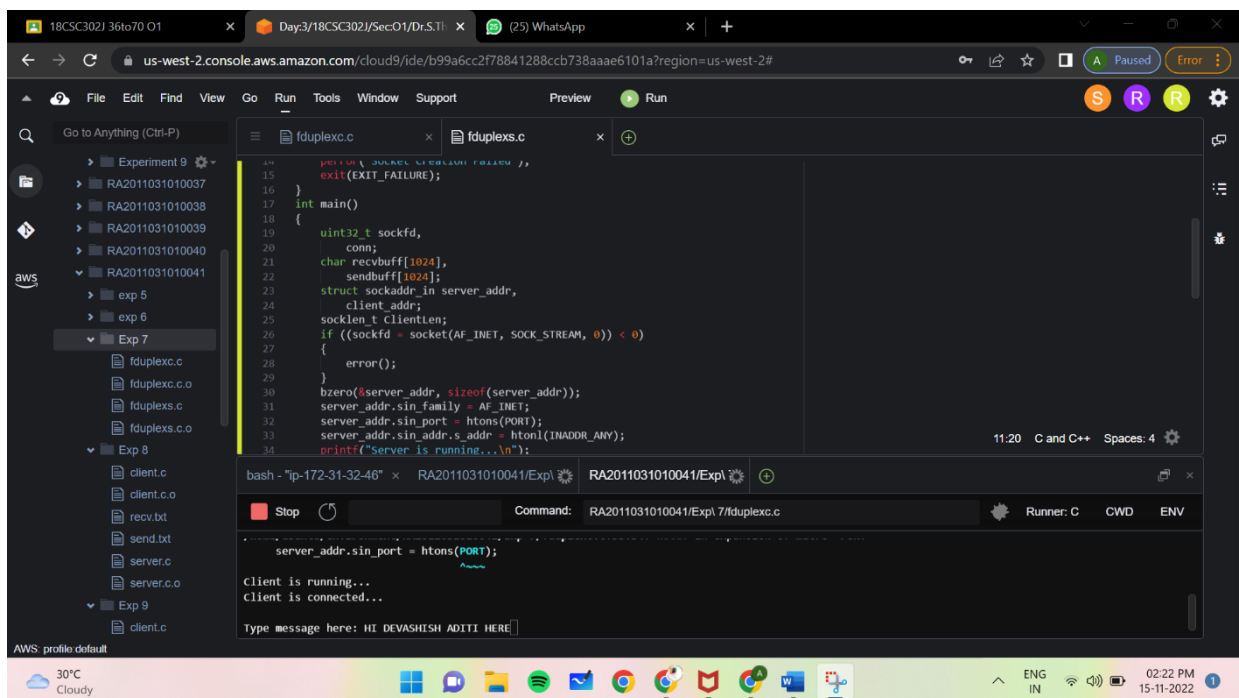
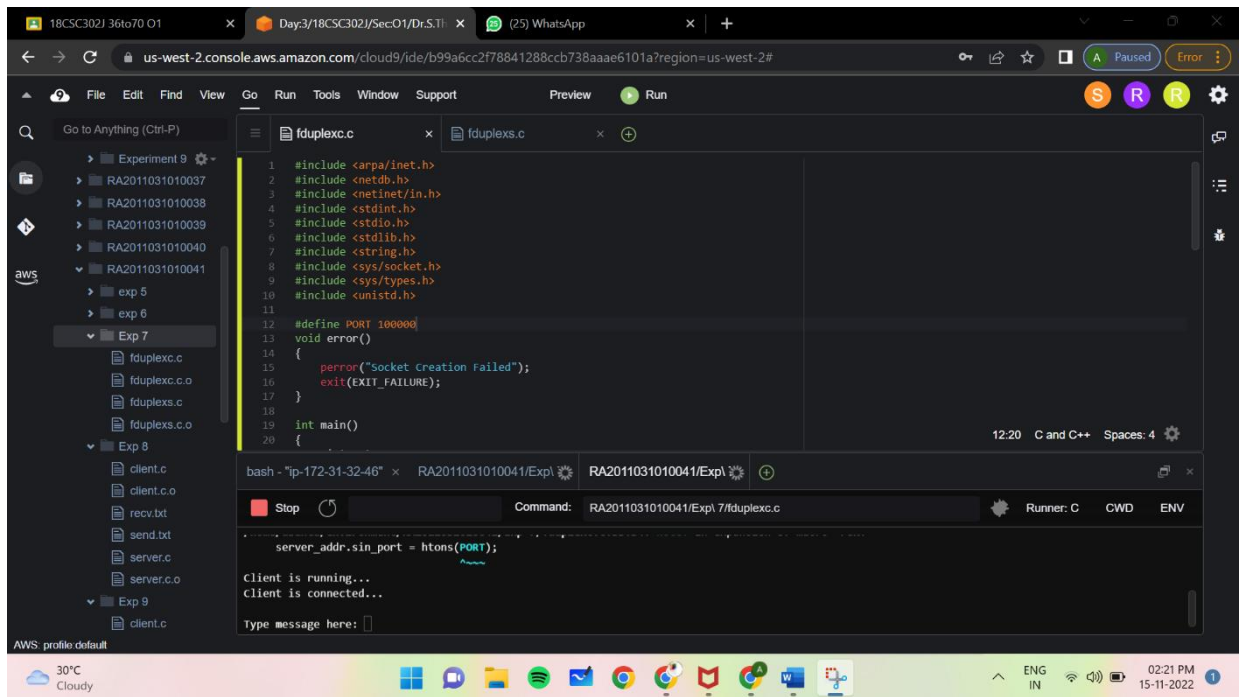
```
1 have a nice day
```

The terminal at the bottom shows the command 'RA2011031010041/Exp1 8/client.c' and its output:

```
bash - "ip-172-31-32-46" x RA2011031010041/Exp1 8/client.c
Run
/usr/include/stdio.h:577:14: note: declared here
extern char *gets (char *__s) __wur __attribute_deprecated__;
/tmp/ccw9DFPe.o: In function 'main':
client.c:(.text+0x0fe): warning: the 'gets' function is dangerous and should not be used.
Enter the port number : 7000
```







CONCLUSION AND FUTURE ENHANCEMENT

The main objective of the project is to develop a Secure Chat Application. We have taken a wide range of literature review in order to achieve all the tasks, where we came to know about some of the products that are existing in the market. We made a detailed research in that path to cover the loop holes that existing systems are facing and to eradicate them in our application. In the process of research We came to know about the latest technologies and different algorithms.

We analyzed various encryption algorithms (DES, AES, IDEA...), Integrity algorithms (MD5, SHA), key-exchange algorithms, authentication and We have implemented those functionalities in my application. We have done a detailed research on Certificate Authority and key tool for the generation of certificates.

The portability of the application has been achieved by using some of the latest JSSE technologies. We implemented these functionalities using JSSE api's. We have gone through core and security concepts of java (JSSE, JCA) packages and for developing GUI We have implemented java swings.

REFERENCES

[1] FTP MODEL IN NESO ACADEMY

[2] FTP MODEL in GeeksforGeeks

<https://www.geeksforgeeks.org/file-transfer-protocol-ftp/amp/>

[3] Multimedia chat working in getstream.io

<https://getstream.io/chat/>

[4] Full Duplex on javatpoint

<https://www.javatpoint.com/types-of-transmission>

[5] Client Server for FTP on www.technopedia.com

<https://www.techopedia.com/definition/7336/file-transfer-protocol-client-ftp-client>
<https://www.techopedia.com/definition/26108/ftp-server>