A Thesis/Project/Dissertation Report

on

CHATTING APPLICATION USING SWING AND NETWORKING

Submitted in partial fulfillment of the requirement for the award of the degree of

B.TECH(CSE)

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Ms. AANCHAL VIJ
ASSISTANT PROFESSOR

Submitted By

AYUSHI PANDEY

19021011481

HARSH JAISWAL

19021180024

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

OCT,2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"CHATTING APPLICATION USING SWING AND NETWORKING"** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of Ms. AANCHAL VIJ ASSISTANT PROFESSOR, Department of Computer Science and Engineering, Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

HARSH JAISWAL 19021180024

AYUSHI PANDEY 19021011481

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Ms. AANCHAL VIJ ASSISTANT PROFESSOR)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **Harsh Jaiswal(19SCSE1180026)** and **Ayushi Pandey(19SCSE1010293)** has been held on 24th December, 2021and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: 24th December, 2021

Place: Greater Noida

Abstract

In todays world everyone in there life is busy in race of achieving goals and for that they need to run a lot from here to there. Also these days, if any big meeting is to be done or any big gatherings are needed to be held then a person will think many times for attending it considering the factors of traffic jam on his way, facilities for having night stays there. Recently whole world is suffering from viruses like corona where it is impossible to have social gatherings. So to overcome these problem we need a platform where people can virtually chat with others. So chat application can be very efficient solution to this problem.

In this project, one can do live chat with others. For this we use server- client architecture. In this, we used to have a server and with this server different clients can connect. Now, in order to establish connection with server with clients weuse the concept of socket programming. It is a desktop based application.

The tools which we are using here are-Language
Used- Java Core
Concept Used- Swing and Socket Programming
IDE Used- Apache NetBeans

If throwing some light on the future of this program, so this project has great scope in future. In future we will include many features to this program.

- Like we can enable username and password method to the code and maintain the database of the clients connection to the server.
- We can increase the capacity of server, so that more number of clients can make connection with the server.
- Features of reacting on particular message needs to be enabled.

List of Tables

List of Figures

Acronyms

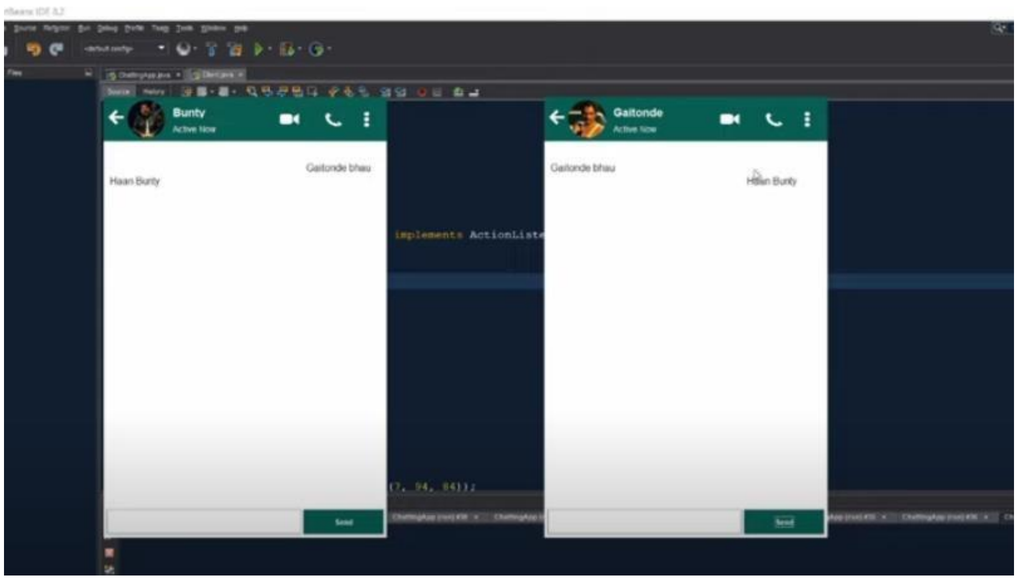
B.Tech.	Bachelor of Technology
M.Tech.	Master of Technology
BCA	Bachelor of Computer Applications
MCA	Master of Computer Applications
B.Sc. (CS)	Bachelor of Science in Computer Science
M.Sc. (CS)	Master of Science in Computer Science
SCSE	School of Computing Science and Engineering

Table of Contents

CHAPTER-1
Introduction

1.1 INTRODUCTION

This project is about how different users can communicate in real time with each. With real time it can said to communicate live with each other on a single platform. So to make this thing possible we have a communication platform for chatting which is client server architecture. In this application, we will have various clients who can join a private or public server and can communicate with other clients using that server. In this world we have different platforms which uses this concept like youtube. In youtube, when someone do live streaming you will see various people are chatting side by side of video in real time. So here there is a server of youtube and we have various users as clients who are connected to that server and it is server-client architecture through which it was possible that all users were able to have a communication with other clients in present time on a single platform. Now in this chat app to communicate through machines of individuals and server we have a thing in each machine known as sockets. These are only sockets through which it was possible for different machines to establish connection with each other. This establishment of different machines is possible only due to socket programming. Due to socket programming it is possible for machines to have end to end communication and send data from one device to other. Now we will see about sockets in detail in system development section. There is one more application where this web chat is brought in great use.



Above is an to demonstrate the use of Desktop Chat. In this example, you can see that we have Web Chat application in which we have different users who are doing real time chat on a common platform which is its server. Each user's machine would be having its socket and server would be having its server through which connection took place between them and they were able to chat with each other. In this

CHAPTER-1
Introduction

application, we would be using the java and socket programming. For sending data from machine M1 to machine M2, Output-stream is used and to receive data from machine M2 to M1 Input-stream is used. In socket programming client should know the IP address of server and port number of applications in order to do distribution ofdata.

CHAPTER-1
Introduction

1.2 Formulation of Problem

In todays world everyone in there life is busy in race of achieving goals and for that they need to run a lot from here to there. Also these days, if any big meeting is to be done or any big gatherings are needed to be held then a person will think many times for attending it considering the factors of traffic jam on his way, facilities for having night stays there.
Recently whole world is suffering from viruses like corona where it is impossible to have social gatherings. So to overcome these problem we need a platform where people can virtually attended meetings and do talking to each other in live. So web chat app can be a very good and efficient solution to this problem. Using socket programming and client-server architecture it is very easy to do remote chatting for users from there homes. But practically it is not possible that everyone has a computer and he can do web chatting. So it is also platform independent. As these days a cheaper smartphone is available with sockets as is capable of doing virtual live chats. So no matter how far you are the only thing required is any type of connection with server and one can use the benefits of web chat.

Now major question is how to achieve this communication. For this we will use client-server architecture approach. In this different clients located at different locations can connect to a common server by knowing the address of server and server will distribute the data of one server with other users. Now working of this chat app will be seen in implementation section.

1.2.1 Tools and Technology Used

Language Used- Java Core
Concept Used- Swing and Socket Programming
IDE Used- Apache NetBea

CHAPTER-2
Literature Survey

These days social networking is very common thing which is performed by people. Social networking is the thing which not only deals with text or sentences but also deals with pictures which we have achieved to operate the picture for performing face detection and finding the expressions. In an organization, colleagues working there can send and get reply of messages instantly in very less time without having face to face conversation, and in the mean time the report of work can be shared/sent instantly during the chat session. With this application it is possible to have virtual conference without getting all the people together in a meeting room physically present. Doing instant messages for company communication is more efficient than making phone calls or doing emails. Various clients can do chatting concurrently. Dependent on a conference call or doing electronic mail message for meeting with colleagues is time consuming, but with this application everybody can join and have a discussion on various topics in very less time. If you really want to have fast communication then this application is far better than e-mails. With the support of fast messages one can send a message and get response of message in very few seconds.

Now, the architecture constitute of client and server module which include the given below steps:

1. Initial step is to execute server program on server machine.
2. After that client send request to server through its device and on that basis server give response to him/her and connection start taking place.
3. When the client-server connection is successfully established, the server then broadcast a list to of its connected users to its each connected client.
4. Client has the right to view all its active users and thus can make communication with them.
5. Server establishes a separate connection for each of its connected client, for which server produces a personal thread for each client connection. Now this thread is responsible to send/receive data from and to clients.
6. Whenever a client creates and sends a message to other client, this message initially is transferred to the server.
7. Then the server transmits the following message to the desired receiver of user.
8. Now, when the receiver client receives the message, receiver can read it.
9. In reflection receiver can send back reply but again follow the process of same process as mentioned above.
10. This chat application brings the use of concept of socket programming and the concept of multithreading. There will be different threads in it. One thread is for running server program and a different thread to control each client that wants to establish connection.

Java Socket Programming

Basically java socket programming is brought in use for communicating between different applications executing on various Java Runtime Environment. Java Socket programming can have two types of mediums to communicate. Communications can be connection-oriented or can be connection-less.
If we talk about connection-oriented sockets then for those we have classes like Socket and Server Socket while, for connection-less socket programming we have Datagram Socket and Datagram packet classes.
The client who are doing socket programming should know two things about server:
1. Client should know the IP Address of Server
2.And Most importantly it should know the port number of server.

Now here we are preparing to establish one-way client and server communication. Taking about this application, client/user transmits a message to the server, then server reads the message using read-line and then prints the message. Here basically we are making use of two classes that includes Server Socket and Socket.
The Socket class is brought in use to communicate between server and client. By using this class, we can do reading and writing of messages. In server program we have server-socket class.
We have a method called accept() in Server Socket class which blocks the console till the time the client is connected. When the client is successfully connected, it brings back the instance of Socket towards side of server. Basically a socket is an endpoint for communications between two devices. To create a socket we have socket class. Various essential methods in socket class which are use includes getinputStream(), getOutputStream(). In server socket class we have socket accept(), synchronised close().

The desktop chat application is based on client - server architecture within a Local Area Network. This client server model of doing computing is basically a distributed application that do pieces of tasks between the provider of resource or we can say that service called servers and service requesters are called clients. The Server side always would be continuously running service listening to a different - different clients enquiring its service. The servers will maintain database of users. Whenever a client makes login to the application, the Server immediately makes authentication of the user . Once the user is been authenticated the IP address of the client/user will registered to the list of the Server and user transmits that list of online client partners and remaining relevant data to the Client/user. When the client wants to have some chat with any other user, then that user's IP address along with his Port address would be sent to the client and vice versa. So a connection would take place and the two clients would be able to chat with each other.

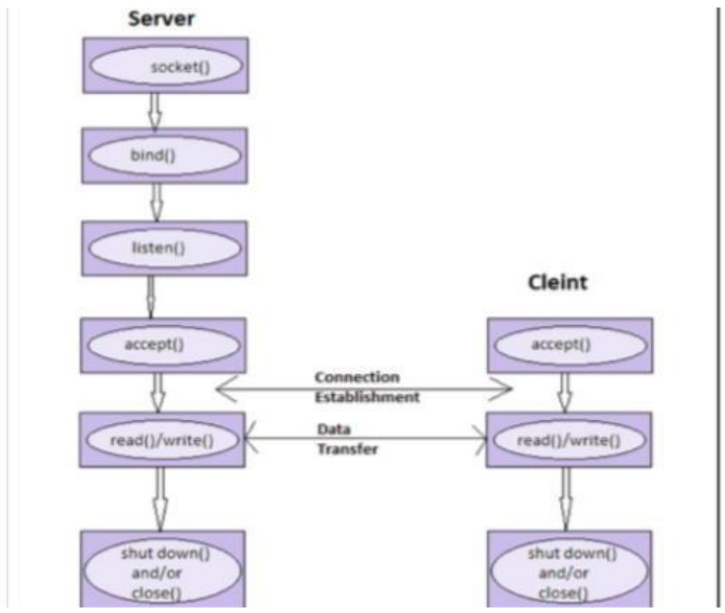


Figure 2

PROJECT DESIGN

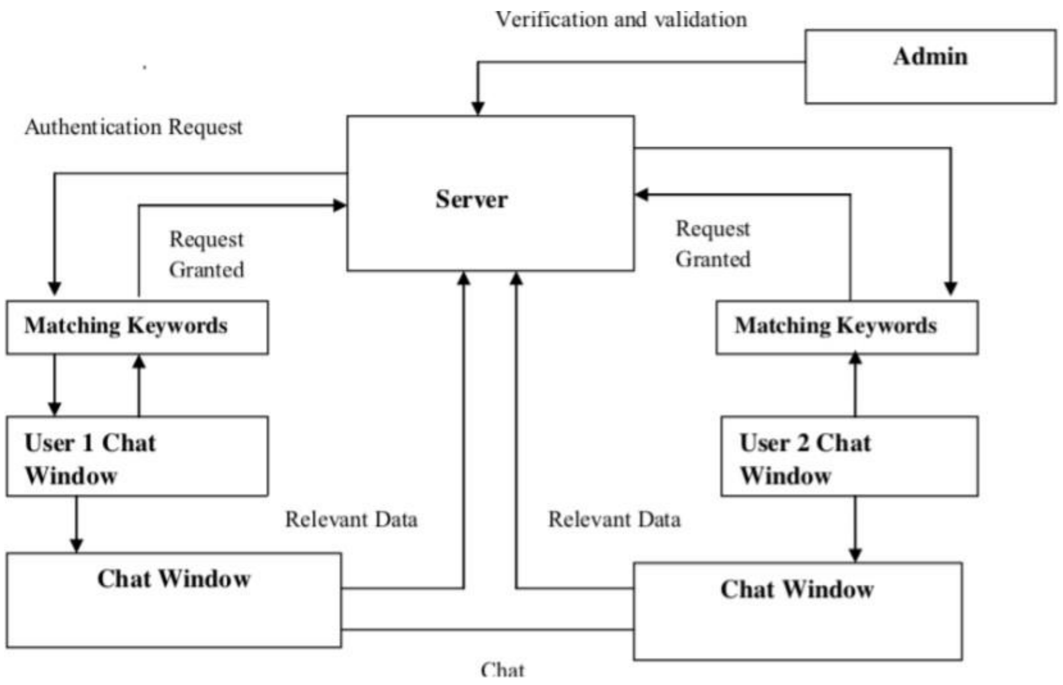


Figure 3

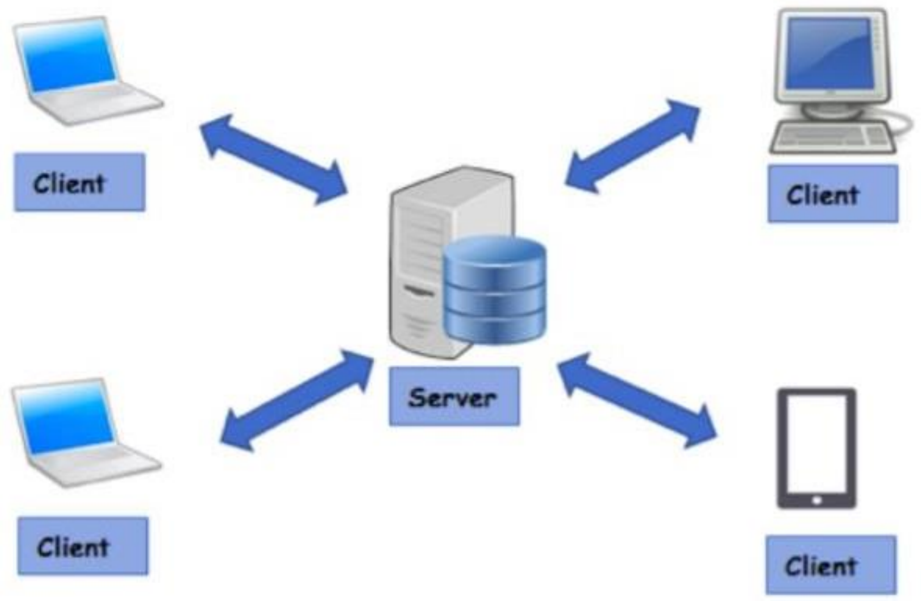


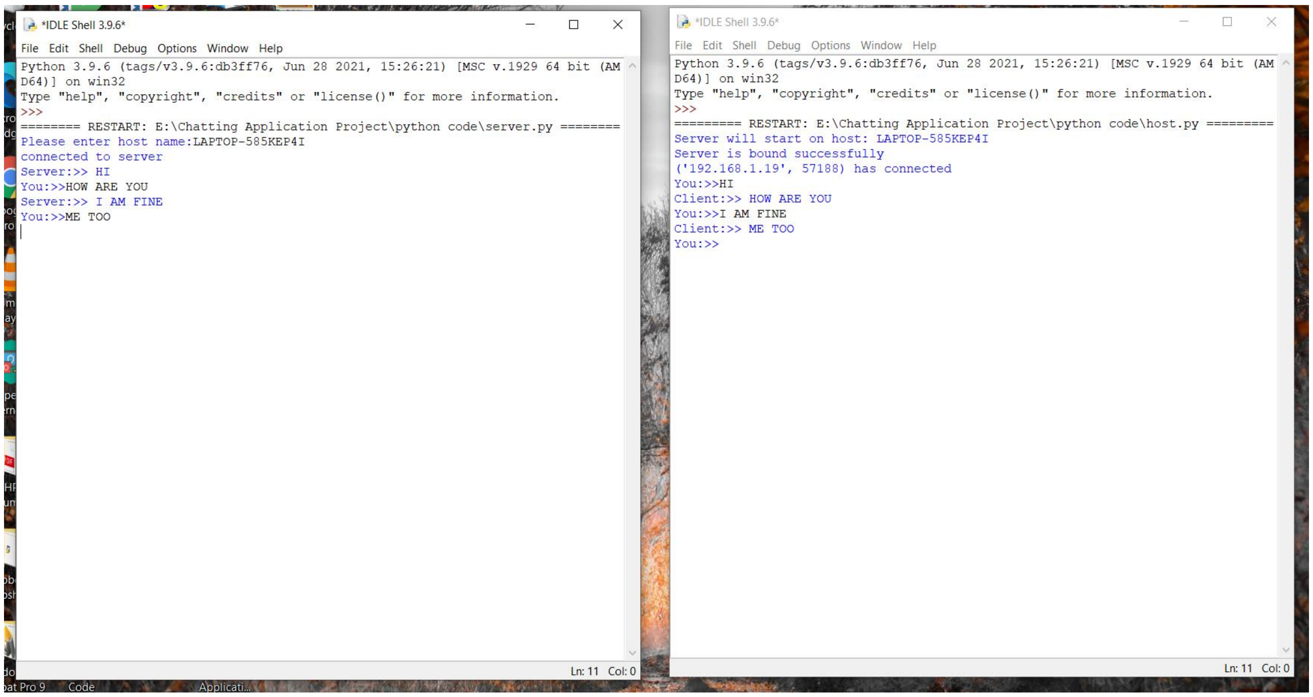
Figure 4

Module Description

<u>Modules of Online Chat Application:</u>

- **Chat Profile Management Module:** Used for managing the Chat Profile details.
- **Multi Chat Module**: Used for managing the details of Multi Chat
- **Users Module**: Used for managing the users of the system

RESULTS



```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: E:\Chatting Application Project\python code\server.py ========
Please enter host name:LAPTOP-585KEP4I
connected to server
Server:>> HI
You:>>HOW ARE YOU
Server:>> I AM FINE
You:>>ME TOO
```

```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: E:\Chatting Application Project\python code\host.py =========
Server will start on host: LAPTOP-585KEP4I
Server is bound successfully
('192.168.1.19', 57188) has connected
You:>>HI
Client:>> HOW ARE YOU
You:>>I AM FINE
Client:>> ME TOO
You:>>
```

CHAPTER- 3

WORKING OF PROJECT

Client- Side Establishment

To connect to another machine we need a socket connection. A socket connection means the two machines have information about each other's network location (IP Address) and TCP port. The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket("127.0.0.1", 5000)
The first argument – IP address of Server. (127.0.0.1 is the IP address of localhost, where code will run on the single stand-alone machine).
The second argument – TCP Port. (Just a number representing which application to run on a server.

For example, HTTP runs on port 80. Port number can be from 0 to 65535)

Communication

To communicate over a socket connection, streams are used to both input and output the data.

Closing the connection

The socket connection is closed explicitly once the message to the server is sent.

In the program, the Client keeps reading input from a user and sends it to the server until "Over" is typed.

```
// A Java program for a Client
import java.net.*;
import java.io.*;

public class Client
{
        // initialize socket and input output streams
        private Socket socket                  = null;
        private DataInputStream input = null;
private DataOutputStream out     = null;

// constructor to put ip address and port
public Client(String address, int port)
{
        // establish a connection
        try
        {
                socket = new Socket(address, port);
                System.out.println("Connected");

                // takes input from terminal
                input = new DataInputStream(System.in);

                // sends output to the socket
                out = new DataOutputStream(socket.getOutputStream());
        }
        catch(UnknownHostException u)
        {
                System.out.println(u);
        }
        catch(IOException i)
        {
                System.out.println(i);
        }

        // string to read message from input
        String line = "";

        // keep reading until "Over" is input
        while (!line.equals("Over"))
        {
                try
                {
                        line = input.readLine();
                        out.writeUTF(line);
                }

                catch(IOException i)
                {
                        System.out.println(i);
```

```
                    }
                }

                // close the connection
                try
                {
                    input.close();
                    out.close();
                    socket.close();
                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
        }

        public static void main(String args[])
        {
                Client client = new Client("127.0.0.1", 5000);
        }
}
```

Server Programming

- A ServerSocket which waits for the client requests (when a client makes a new Socket())
- A plain old Socket socket to use for communication with the client.

Communication
getOutputStream() method is used to send the output through the socket.

Close the Connection
After finishing, it is important to close the connection by closing the socket as well as input/output streams.

```
// A Java program for a Server
    import java.net.*;
    import java.io.*;

    public class Server
    {
        //initialize socket and input stream
        private Socket          socket = null;
        private ServerSocket server = null;
        private DataInputStream in     = null;

        // constructor with port
        public Server(int port)
        {
            // starts server and waits for a connection
            try
            {
                server = new ServerSocket(port);
                System.out.println("Server started");

                System.out.println("Waiting for a client ...");

                socket = server.accept();
                System.out.println("Client accepted");

                // takes input from the client socket
                in = new DataInputStream(
                    new BufferedInputStream(socket.getInputStream()));

                String line = "";

                // reads message from client until "Over" is sent
                while (!line.equals("Over"))
                {
                    try
                    {
                        line = in.readUTF();
                        System.out.println(line);
                        }
                        catch(IOException i)
                        {
                                System.out.println(i);
                        }
                }
                System.out.println("Closing connection");

                // close connection
                socket.close();
                in.close();
        }
        catch(IOException i)
        {
                System.out.println(i);
        }
    }

    public static void main(String args[])
    {
        Server server = new Server(5000);
    }
}
```

- Server application makes a ServerSocket on a specific port which is 5000. This starts our Server listening for client requests coming in for port 5000.
- Then Server makes a new Socket to communicate with the client.

```
        socket = server.accept()
```

- The accept() method blocks(just sits there) until a client connects to the server.

- Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends “Over”.
- After we’re done we close the connection by closing the socket and the input stream.
- To run the Client and Server application on your machine, compile both of

them. Then first run the server application and then run the Client application.

Group Chat Application

```
import java.net.*;
import  java.io.*;
import java.util.*;
public class GroupChat
{
        private static final String TERMINATE = "Exit";
        static String name;
        static volatile boolean finished = false;
        public static void main(String[] args)
        {
                if (args.length != 2)
                        System.out.println("Two arguments required: <multicast-host>
<port-number>");
                else
                {
                        try
                        {
                                InetAddress group = InetAddress.getByName(args[0]);
                                int port = Integer.parseInt(args[1]);
                                Scanner sc = new Scanner(System.in);
                                System.out.print("Enter your name: ");
                                name = sc.nextLine();
                                MulticastSocket socket = new MulticastSocket(port);

                                // Since we are deploying
                                socket.setTimeToLive(0);
                                //this on localhost only (For a subnet set it as 1)

                                socket.joinGroup(group);
                                Thread t = new Thread(new
                                ReadThread(socket,group,port));

                                // Spawn a thread for reading messages
                                t.start();
```

```
                                        // sent to the current group
                                        System.out.println("Start typing messages...\n");
                                        while(true)
                                        {
                                                String message;
                                                message = sc.nextLine();

                if(message.equalsIgnoreCase(GroupChat.TERMINATE))
                                                {
                                                        finished = true;
                                                        socket.leaveGroup(group);
                                                        socket.close();
                                                        break;
                                                }
                                                message = name + ": " + message;
                                                byte[] buffer = message.getBytes();
                                                DatagramPacket datagram = new
                                                DatagramPacket(buffer,buffer.length,group,port);
                                                socket.send(datagram);
                                        }
                                }
                                catch(SocketException se)
                                {
                                        System.out.println("Error creating socket");
                                        se.printStackTrace();
                                }
                                catch(IOException ie)
                                {
                                        System.out.println("Error reading/writing from/to
socket");
                                        ie.printStackTrace();
                                }
                        }
                }
}


class ReadThread implements Runnable
{
        private MulticastSocket socket;
        private InetAddress group;
        private int port;
        private static final int MAX_LEN = 1000;
        ReadThread(MulticastSocket socket,InetAddress group,int port)
        {
            this.socket = socket;
            this.group = group;
            this.port = port;
        }

        @Override
        public void run()
        {
            while(!GroupChat.finished)
            {
                    byte[] buffer = new byte[ReadThread.MAX_LEN];
                    DatagramPacket datagram = new
                    DatagramPacket(buffer,buffer.length,group,port);
                    String message;
                try
                {
                    socket.receive(datagram);
                    message = new
                    String(buffer,0,datagram.getLength(),"UTF-8");
                    if(!message.startsWith(GroupChat.name))
                        System.out.println(message);
                }

catch(IOException e)

{

System.out.println("Socke
t closed!");

}

}

}

}
```

FUTURE SCOPE

With the knowledge we have gained by developing this application, we are confident that in the future we can make the application more effectively by adding this services.

- Extending this application by providing Authorisation service.
- Creating Database and maintaining users.
- Increasing the effectiveness of the application by providing Voice Chat.
- Extending it to Web Support.

Conclusion

There is always a room for improvements in any apps. Right now, we are just dealing with text communication. There are several android apps which serve similar purpose as this project, but these apps were rather difficult to use and provide confusing interfaces. A positive first impression is essential in human relationship as well as in human computer interaction. This project hopes to develop a chat service Android app with high quality user interface. In future we may be extended to include features such as:

1. CSS to beautify the project
2. File Transfer

REFERENCES

- www.wikipedia.com
- www.forbes.com