



A Project Report

On

# **Chat Application**

By

**Shreya Gupta**

**Bsc.IT**

**Batch: - 2020 – 5072**

**Center:- Thane, Mumbai**

Under the Guidance of,

**Jayant V.**

**Technical Trainer**

**EduBridge**

**(School of coding)**

# Introduction:

My project explains about A Chat Application. This project mainly explains the chat between multiple users. This project shows some ease in logging in to the application by multiple users and having a successful discussion among them.

Our Project Includes:-

- User Login/Logout Module
- The Server Side
- The Client Chat windows

I have developed this Application in Java Swing and Java socket programming. It's a desktop based project so I have used a jar file to open the chat window for the client.

The main feature of the project is to have a secured conversation between the clients who have logged in to the chat window.

In the Socket programming, I have made a server side which is firstly executed to start the connection for the clients to login to the chat window.

Using Swing, I have created the GUI of the application for the client's side chat window.

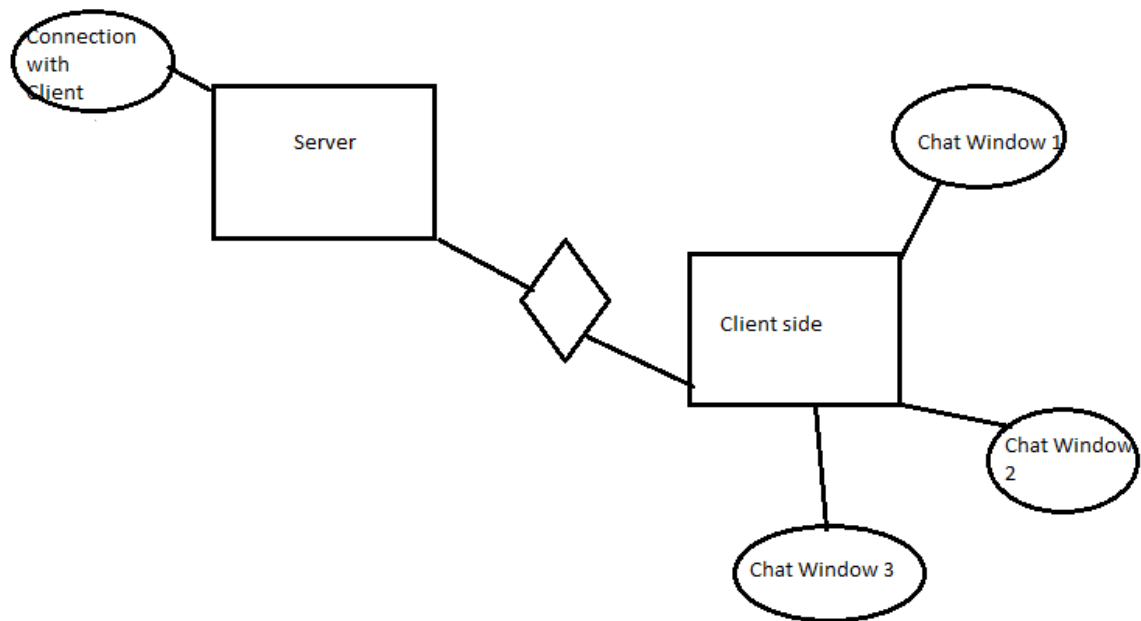
## Software Requirements:

Front end:                      Java SE 8 or above.

Middleware/Server:    Command prompt

Operating System:    Window XP (Minimum).

# ERD (Entity Relationship Diagram):



## MODULES

1. Server
2. Client.jar
3. Chat windows
4. Login/ Logout of users
5. About the Chat

## **Server Module**

The server module comprises of all the coding of the application. We need to run the server.java file on command prompt to start the connection between the client and the server. The GUI has been designed using Java Swing which is a very efficient and a great light weighted tool for creating any GUI. For establishing the connection, Socket programming has been used. As we run the Server file, the connection gets activated. Next, we need to double click on the Client jar file. The client jar has been designed for the chat windows. It has all the functionalities of the GUI and the socket connection established by the server. As we double click on the jar file, a chat window opens. Similarly, again we can double click on the jar file for more chat windows. It has some buttons, details of the login, logout date,time of the user,a scroll bar so that we can scroll and go through the messages send by the various users.

### **Client.jar**

The Client.jar file has all the work to do! We need to double click on the jar file to open the chat window and have some conversations. It has a simple GUI based on Java Swing and connected to the server. We can have as many chat windows we want. It has the login-logout procedure to start chatting. On the right handside, there's the list of the users whoever is available to chat. It shows the names of the users who have logged in one after the other.

### **Login/Logout**

In the application, after opening the chat window, firstly it is required to login to the chat window with your name. As we open the chat window, there is a button saying to login and we need to login to the chat window. After that, in the text area we can see the details related to the users name, date,time and when did the user logout after the chat.

## About the chat!

The project is all about the chat. The application gives us an idea of how we can have different windows and have a conversation on it. We can have any formal or informal kinds of chats between the users. This is a small definition of developing a chat application. We can develop it and make it even more interesting.

## Coding

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class MyServer
{
    ArrayList al=new ArrayList();
    ArrayList users=new ArrayList();
    ServerSocket ss;
    Socket s;

    public final static int PORT=10;
    public final static String UPDATE_USERS="updateuserslist:";
    public final static String LOGOUT_MESSAGE="@ @logoutme@ @:";
    public MyServer()
    {
        try{
            ss=new ServerSocket(PORT);
            System.out.println("Server Started "+ss);
            while(true)
            {
                s=ss.accept();
                Runnable r=new MyThread(s,al,users);
                Thread t=new Thread(r);
                t.start();
            }
            // System.out.println("Total alive clients : "+ss.);
        }
    }
}
```

```

    }
    catch(Exception e){System.err.println("Server constructor"+e);}
}

public static void main(String [] args)
{
    new MyServer();
}

}

class MyThread implements Runnable
{
    Socket s;
    ArrayList al;
    ArrayList users;
    String username;

    MyThread (Socket s, ArrayList al,ArrayList users)
    {
        this.s=s;
        this.al=al;
        this.users=users;
        try{
            DataInputStream dis=new DataInputStream(s.getInputStream());
            username=dis.readUTF();
            al.add(s);
            users.add(username);
            tellEveryone("***** "+ username+" Logged in at "+(new
Date())+" *****");
            sendNewUserList();
        }
        catch(Exception e){System.err.println("MyThread constructor "+e);}
    }

    public void run()
    {
        String s1;
        try{

```

```

        DataInputStream dis=new DataInputStream(s.getInputStream());
        do
        {
            s1=dis.readUTF();
            if(s1.toLowerCase().equals(MyServer.LOGOUT_MESSAGE))
break;
//    System.out.println("received from "+s.getPort());
        tellEveryOne(username+" said: "+" : "+s1);
        }
        while(true);
        DataOutputStream tdos=new
DataOutputStream(s.getOutputStream());
        tdos.writeUTF(MyServer.LOGOUT_MESSAGE);
        tdos.flush();
        users.remove(username);
        tellEveryOne("***** "+username+" Logged out at "+(new
Date())+" *****");
        sendNewUserList();
        al.remove(s);
        s.close();

    }
catch(Exception e){System.out.println("MyThread Run"+e);}
}

public void sendNewUserList()
{
    tellEveryOne(MyServer.UPDATE_USERS+users.toString());

}

public void tellEveryOne(String s1)
{
    Iterator i=al.iterator();
    while(i.hasNext())
    {
        try{
            Socket temp=(Socket)i.next();
            DataOutputStream dos=new

```

```

DataOutputStream(temp.getOutputStream());
    dos.writeUTF(s1);
    dos.flush();
    //System.out.println("sent to : "+temp.getPort()+" : "+ s1);
}
catch(Exception e){System.err.println("TellEveryone "+e);}
}
}

}

```

class MyClient implements ActionListener

```

{
Socket s;
DataInputStream dis;
DataOutputStream dos;

```

```

JButton sendButton, logoutButton,loginButton, exitButton;
JFrame chatWindow;
JTextArea txtBroadcast;
JTextArea txtMessage;
JList usersList;

```

```

public void displayGUI()
{
chatWindow=new JFrame();
txtBroadcast=new JTextArea(5,30);
txtBroadcast.setEditable(false);
txtMessage=new JTextArea(2,20);
usersList=new JList();

```

```

sendButton=new JButton("Send");
logoutButton=new JButton("Log out");
loginButton=new JButton("Log in");
exitButton=new JButton("Exit");

```

```

JPanel center1=new JPanel();
center1.setLayout(new BorderLayout());

```



```
center1.add(new JLabel("Broad Cast messages from all online  
users",JLabel.CENTER),"North");  
center1.add(new JScrollPane(txtBroadcast),"Center");
```

```
JPanel south1=new JPanel();  
south1.setLayout(new FlowLayout());  
south1.add(new JScrollPane(txtMessage));  
south1.add(sendButton);
```

```
JPanel south2=new JPanel();  
south2.setLayout(new FlowLayout());  
south2.add(loginButton);  
south2.add(logoutButton);  
south2.add(exitButton);
```

```
JPanel south=new JPanel();  
south.setLayout(new GridLayout(2,1));  
south.add(south1);  
south.add(south2);
```

```
JPanel east=new JPanel();  
east.setLayout(new BorderLayout());  
east.add(new JLabel("Online Users",JLabel.CENTER),"East");  
east.add(new JScrollPane(usersList),"South");
```

```
chatWindow.add(east,"East");
```

```
chatWindow.add(center1,"Center");  
chatWindow.add(south,"South");
```

```
chatWindow.pack();  
chatWindow.setTitle("Login for Chat");  
chatWindow.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);  
chatWindow.setVisible(true);  
sendButton.addActionListener(this);  
logoutButton.addActionListener(this);  
loginButton.addActionListener(this);  
exitButton.addActionListener(this);
```

```

logoutButton.setEnabled(false);
loginButton.setEnabled(true);
txtMessage.addFocusListener(new FocusAdapter()
{public void focusGained(FocusEvent fe){txtMessage.selectAll();}});

chatWindow.addWindowListener(new WindowAdapter()
{
public void windowClosing(WindowEvent ev)
{
if(s!=null)
{
JOptionPane.showMessageDialog(chatWindow,"You are logged in right
now :) ","Exit",JOptionPane.INFORMATION_MESSAGE);
logoutSession();
}
System.exit(0);
}
});
}

public void actionPerformed(ActionEvent ev)
{
JButton temp=(JButton)ev.getSource();
if(temp==sendButton)
{
if(s==null)
{JOptionPane.showMessageDialog(chatWindow,"You have not
logged in, Please login first!"); return;}
try{
dos.writeUTF(txtMessage.getText());
txtMessage.setText("");
}
catch(Exception excp){txtBroadcast.append("\nsend button click
:"+excp);}
}
if(temp==loginButton)
{
String uname=JOptionPane.showInputDialog(chatWindow,"Enter Your
Sweet Name: ");

```

```

if(uname!=null)
    clientChat(uname);
}
if(temp==logoutButton)
{
if(s!=null)
    logoutSession();
}
if(temp==exitButton)
{
if(s!=null)
{
JOptionPane.showMessageDialog(chatWindow,"Oops! You are logged
out. ","Exit",JOptionPane.INFORMATION_MESSAGE);
logoutSession();
}
System.exit(0);
}
}

```

```

public void logoutSession()
{
if(s==null) return;
try{
dos.writeUTF(MyServer.LOGOUT_MESSAGE);
Thread.sleep(500);
s=null;
}
catch(Exception e){txtBroadcast.append("\n inside logoutSession
Method"+e);}
}

```

```

logoutButton.setEnabled(false);
loginButton.setEnabled(true);
chatWindow.setTitle("Login for Chat");
}

```

```

public void clientChat(String uname)
{
try{

```

```

s=new Socket(InetAddress.getLocalHost(),MyServer.PORT);
dis=new DataInputStream(s.getInputStream());
dos=new DataOutputStream(s.getOutputStream());
ClientThread ct=new ClientThread(dis,this);
Thread t1=new Thread(ct);
t1.start();
dos.writeUTF(uname);
chatWindow.setTitle(uname+" Chat Window");
}
catch(Exception e){txtBroadcast.append("\nClient Constructor " +e);}
logoutButton.setEnabled(true);
loginButton.setEnabled(false);
}

public MyClient()
{
    displayGUI();
    //clientChat();
}

public static void main(String []args)
{
    new MyClient();
}

}

class ClientThread implements Runnable
{
    DataInputStream dis;
    MyClient client;

    ClientThread(DataInputStream dis,MyClient client)
    {
        this.dis=dis;
        this.client=client;
    }

```

```

public void run(){
String s2="";
do
{
    try{
        s2=dis.readUTF();
        if(s2.startsWith(MyServer.UPDATE_USERS))
            updateUserList(s2);
        else if(s2.equals(MyServer.LOGOUT_MESSAGE))
            break;
        else
            client.txtBroadcast.append("\n"+s2);
        int
lineOffset=client.txtBroadcast.getLineStartOffset(client.txtBroadcast.getL
ineCount()-1);
        client.txtBroadcast.setCaretPosition(lineOffset);
    }
    catch(Exception e){client.txtBroadcast.append("\nClientThread run
: "+e);}
}
while(true);
}

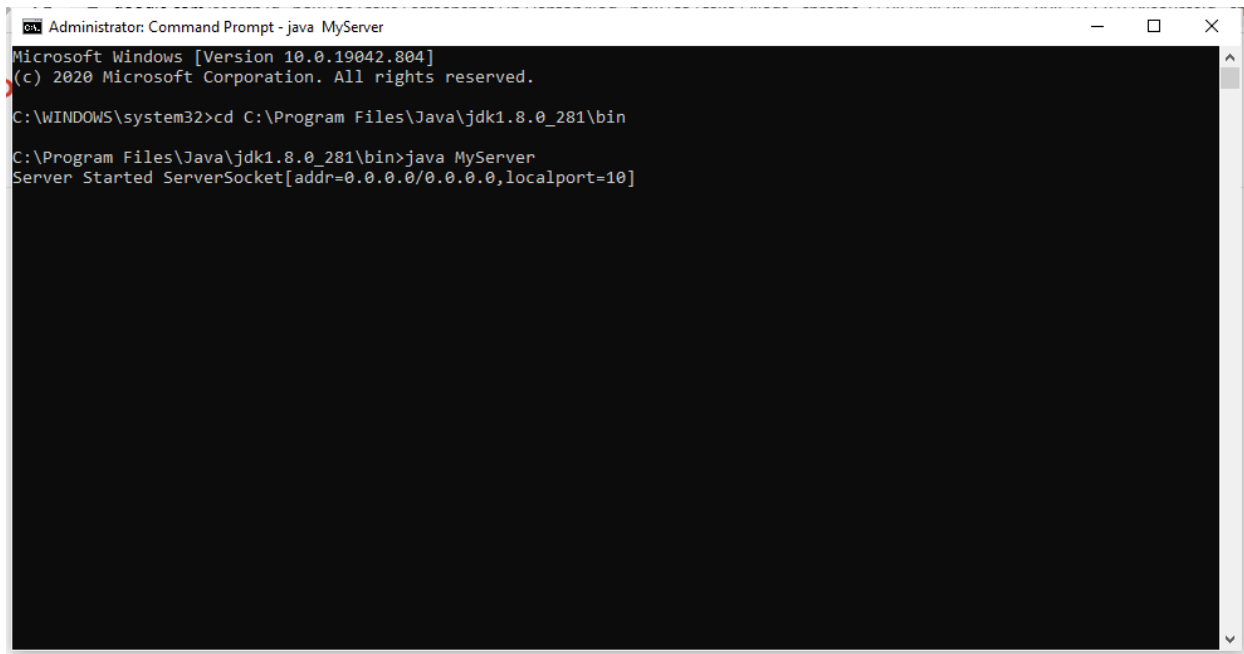
public void updateUserList(String ul)
{
Vector ulist=new Vector();

ul=ul.replace("[","");
ul=ul.replace("]", "");
ul=ul.replace(MyServer.UPDATE_USERS,"");
StringTokenizer st=new StringTokenizer(ul,"");

while(st.hasMoreTokens())
{
String temp=st.nextToken();
ulist.add(temp);
}
client.usersList.setListData(ulist);
}}

```

# Screenshots of the project:



```
Administrator: Command Prompt - java MyServer
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Program Files\Java\jdk1.8.0_281\bin

C:\Program Files\Java\jdk1.8.0_281\bin>java MyServer
Server Started ServerSocket[addr=0.0.0.0/0.0.0.0,localport=10]
```

