

Advanced Computer Networks

Project Report

Name - Rahul Deepak Prabhu

NET ID - rdp130130

Name - ShubhankarVenkatesh

NET ID - sxv148130

Problem Statement: Chat Program: Design and program (socket programming) a simple chat session application. Design a session between two individuals on separate computers to exchange messages. (An extended version of this project, which include graphical interface, may be a 2-person project).

For Extra Credit - We have implemented multi-user chat session as well as Android Application for Cross Platform Chat session, i.e. between PCs and Android Phones. We have also implemented the chat session with time stamps for each message.

Introduction:

Our objective was to create a chat session between two or more individuals in a graphical interface from different machines. The chat session was to be based on socket programming. We used Java language to implement the graphical chat session based on socket programming. Considering there are a number of graphical modules are present in Java which can be easily modified to our own requirement. Hence we had more inclination of implementing the chat session in Java. To add more flexibility of adding any number of users to this session we decided to setup a server of our own using Amazon AWS Cloud. With this step we were able to add any number of users in the chat session. The user would just have to run the client program, enter the server IP address and start their chat sessions after entering their screen name to be displayed for identification. When the client is registered with a unique name, then the user can begin their chat sessions and their messages will be broadcast to all the active users. We have also implemented Android Application for the Chat Program, thus we were able to integrate the chat program onto PC as well as Android Phones

Problem Breakdown:

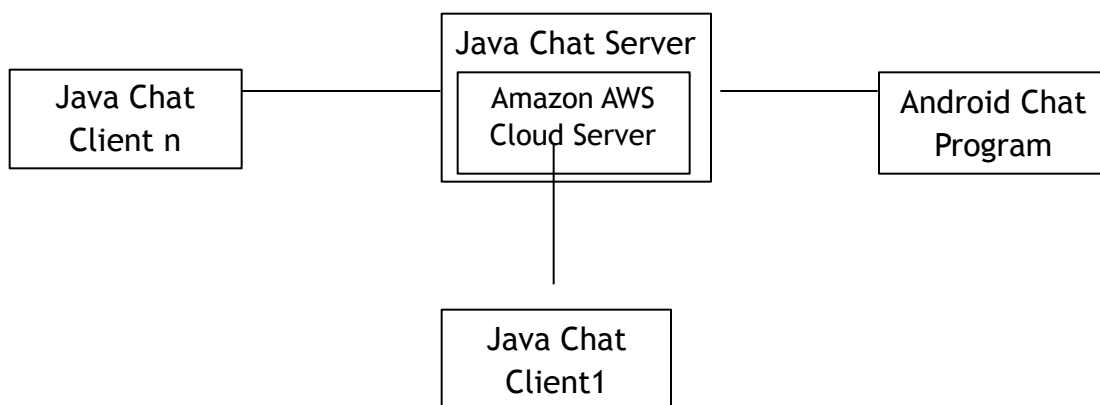
Chat Server:

1. To make chat session using socket programming, we used the "Socket" class of java to implement a TCP socket connection between the Chat Server and the Chat Client.
2. The java server was kept in infinite loop to keep listening to the Port "9001" for any incoming connection.
3. Since the problem of chat session was to implement chat session on 3 or more different machines, hence uniqueness of user logins was thought of. Hence we build in an exception handler to discard any request for chat login with repeated name.
4. After the user has been accepted, the server would read the input stream and broadcast it to all the currently active clients. *The previous chat conversations will be unavailable*.

Chat Client:

1. For GUI interface of Chat Windows, we used Java Swing class to implement chat window boxes.
2. To start chat session the user would be required to enter the IP address of the server and server would request for Screen Name to be used for Chat Session.
3. To implement uniqueness of client, the chat server would run exception handler until the user enters a unique screen name. The chat window will be enabled only when the screen name of the user is accepted by the server.
4. After user is accepted, the stream reader will read the text entered by the user and it will be broadcast to all other chat clients along with the timestamp.

Block Diagram



Problems Faced: We had many problems that we faced when we were implementing the client server:

- We did not have all the libraries required as we had an older version of eclipse(IDE for java)
- With the android app we could not run the app on the phone but it worked on the emulator. We had to initialize a new UI thread every time we had to play with the ui.

```

public void clearentername(){
    runOnUiThread(new Runnable() {
        public void run(){
            EditText text = (EditText)findViewById(R.id.entername);
            text.setText(nil);});
}

```

- Permissions for the internet was not enabled by default for the app and we had to enable it through the manifest file for the app

```

5      android:versionName="1.0" />
6
7      <uses-sdk
8          android:minSdkVersion="14"
9          android:targetSdkVersion="21" />
10     <uses-permission android:name="android.permission.INTERNET"/>
11     <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
12     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
13     <uses-permission android:name="android.permission.READ_USER_DICTIONARY"/>
14     <application
15         android:allowBackup="true"

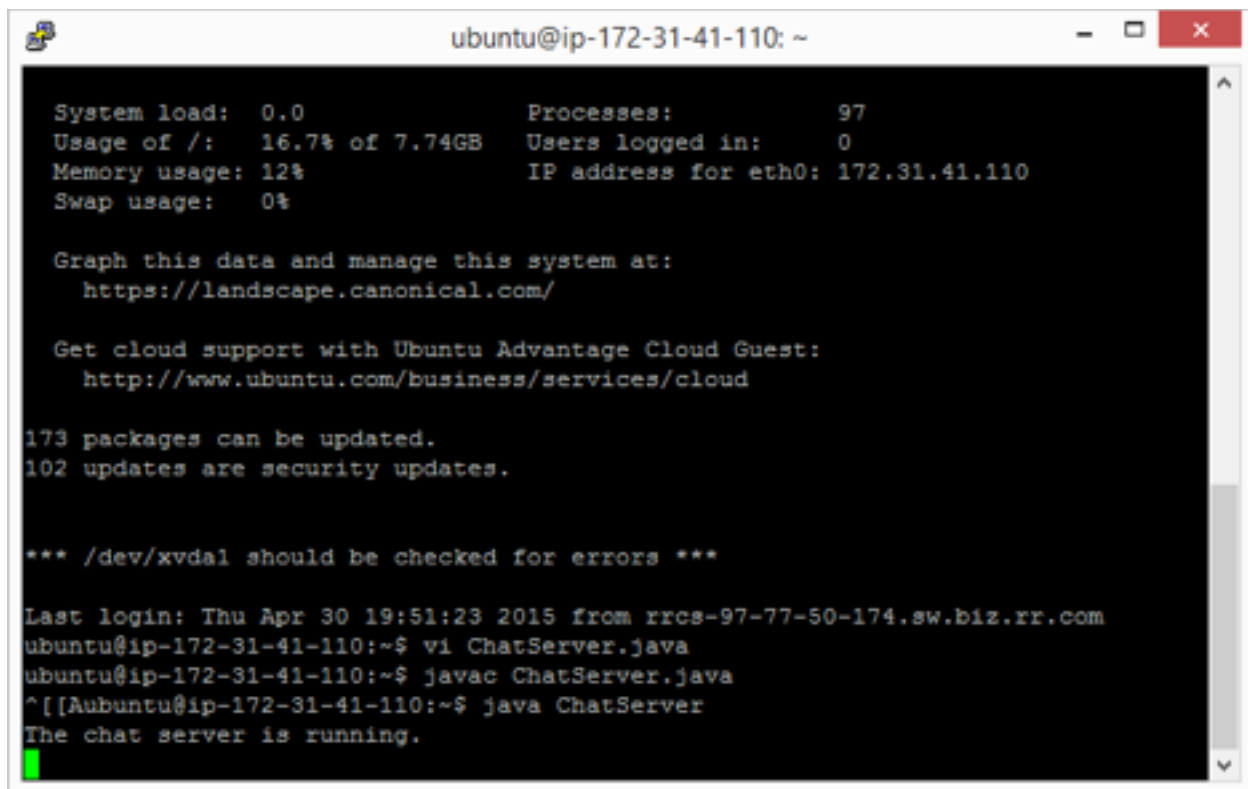
```

- Debugging android at runtime is very tedious as we had to go through the logcat at every instance of time
- We encountered many other problems but we could clear them through many online forums.
- We have all the code on github for further development.

Conclusion:

The multi-user chat session was successfully implemented using Java programming language in a graphical interface with user on 3 or more machines, as well as Android N-Way Chat Session Application. Overall we were able to develop a Multi-user cross platform chat messenger for the project.

Screen Shots



A terminal window titled 'ubuntu@ip-172-31-41-110: ~' displays system statistics and the execution of a Java chat server. The statistics include system load (0.0), memory usage (12%), and IP address (172.31.41.110). It also shows that 173 packages can be updated, with 102 being security updates. The user runs 'vi ChatServer.java', 'javac ChatServer.java', and 'java ChatServer', resulting in the message 'The chat server is running.'.

```
ubuntu@ip-172-31-41-110: ~  
  
System load: 0.0          Processes:          97  
Usage of /:  16.7% of 7.74GB Users logged in:   0  
Memory usage: 12%        IP address for eth0: 172.31.41.110  
Swap usage:  0%  
  
Graph this data and manage this system at:  
  https://landscape.canonical.com/  
  
Get cloud support with Ubuntu Advantage Cloud Guest:  
  http://www.ubuntu.com/business/services/cloud  
  
173 packages can be updated.  
102 updates are security updates.  
  
*** /dev/xvda1 should be checked for errors ***  
  
Last login: Thu Apr 30 19:51:23 2015 from rrcs-97-77-50-174.sw.biz.rr.com  
ubuntu@ip-172-31-41-110:~$ vi ChatServer.java  
ubuntu@ip-172-31-41-110:~$ javac ChatServer.java  
^[[Aubuntu@ip-172-31-41-110:~$ java ChatServer  
The chat server is running.  
█
```

Fig 1 - Chat Server Running on Amazon EC2 Server with IP address

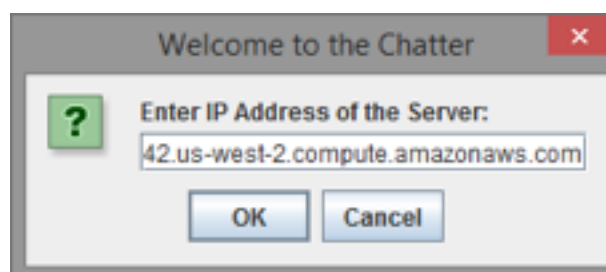


Fig 2 - Chat User Entering Chat Server IP Address

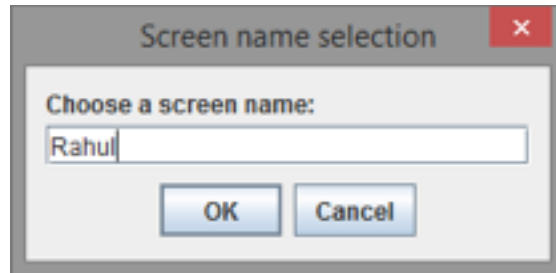


Fig 3 - Entering Screen Name of Chat User for Identification

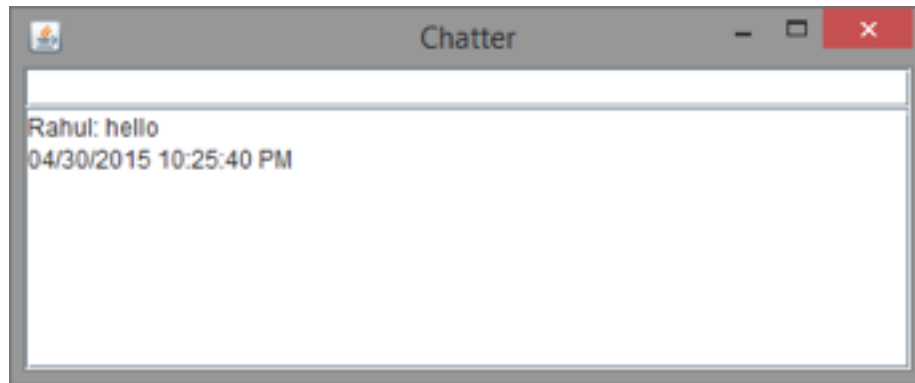


Fig 4 - Chat Messages Along with Timestamp

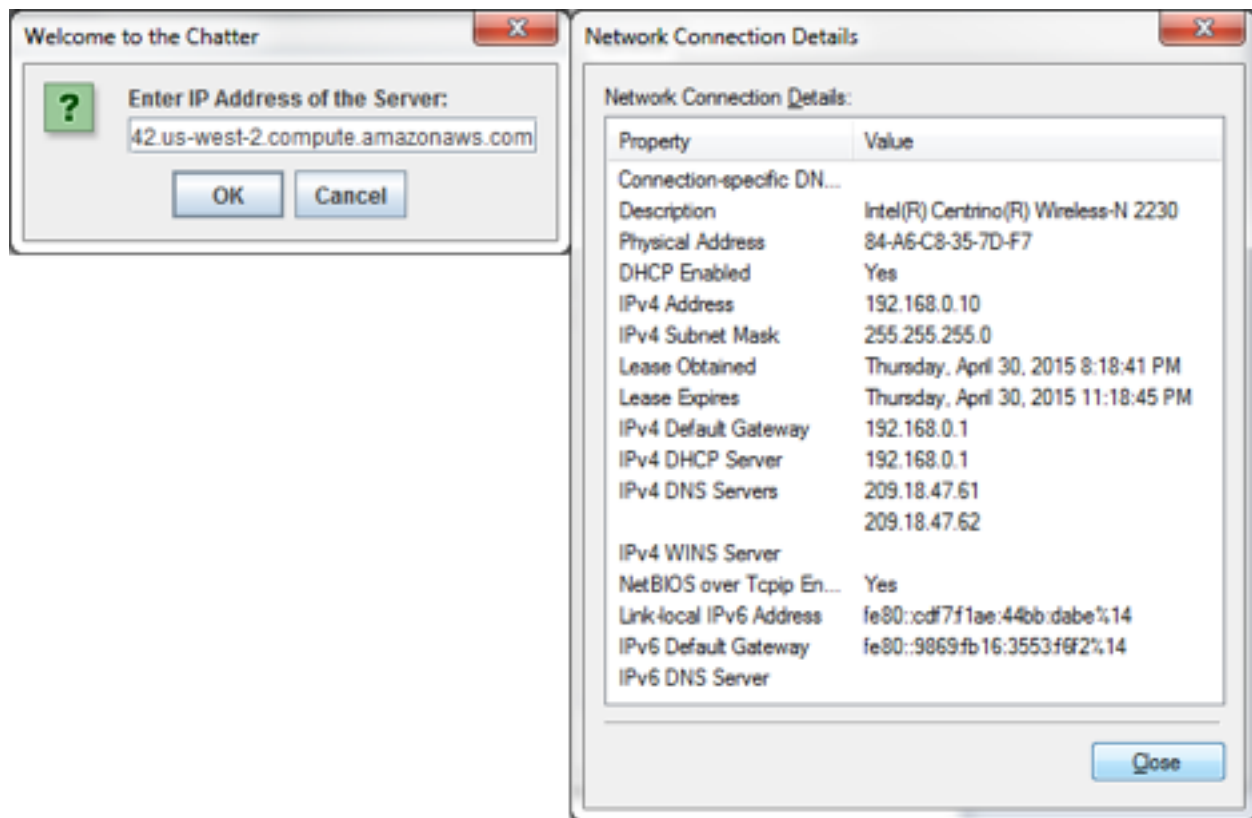


Fig 5 - User 2 Entering Logging Onto Chat Session with Different IP

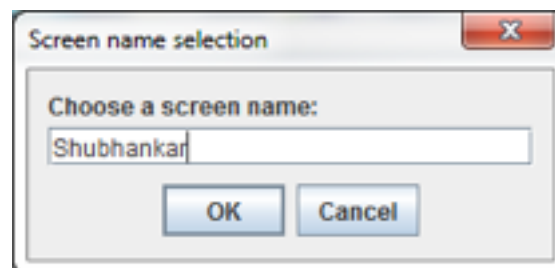


Fig 6 - User 2 Entering Name for Starting Chat Session

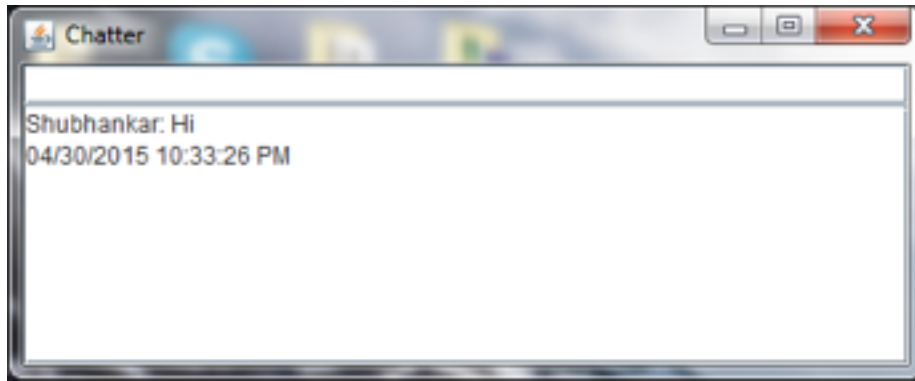


Fig 7 - User 2's Chat Message Window

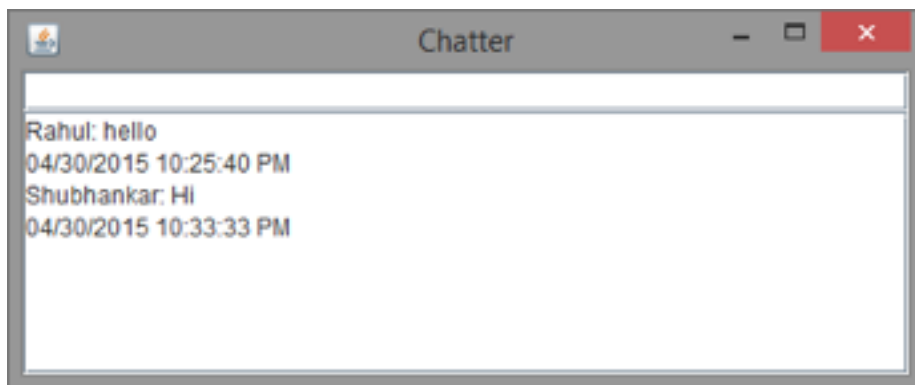


Fig 8 - User 1's Chat Message Window after 2nd User Messaging



Fig 9 - Opening Chat Session on Android

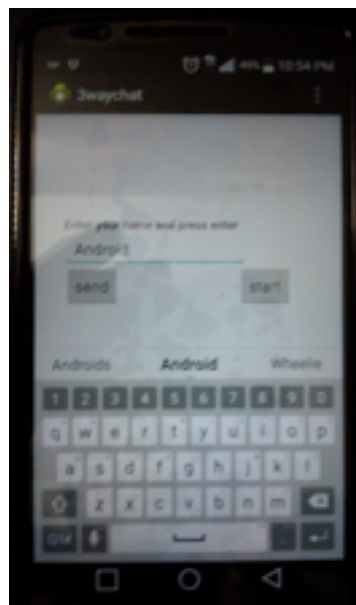


Fig 10 - Entering Username in Android Chat Session

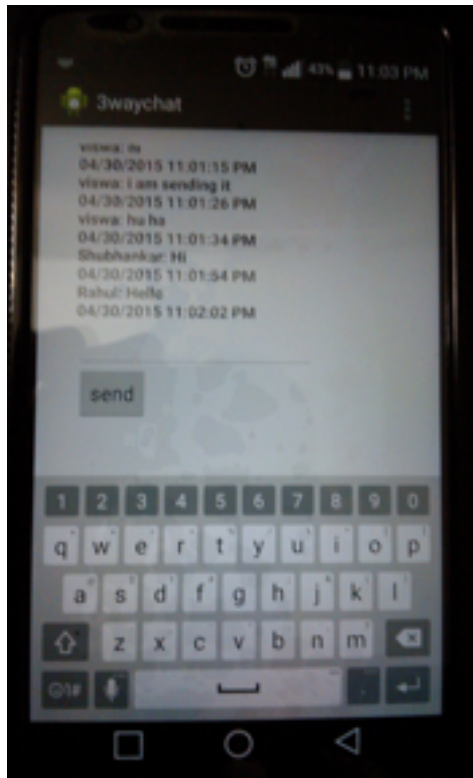


Fig 11 - Chat Messages in Android Chat Window

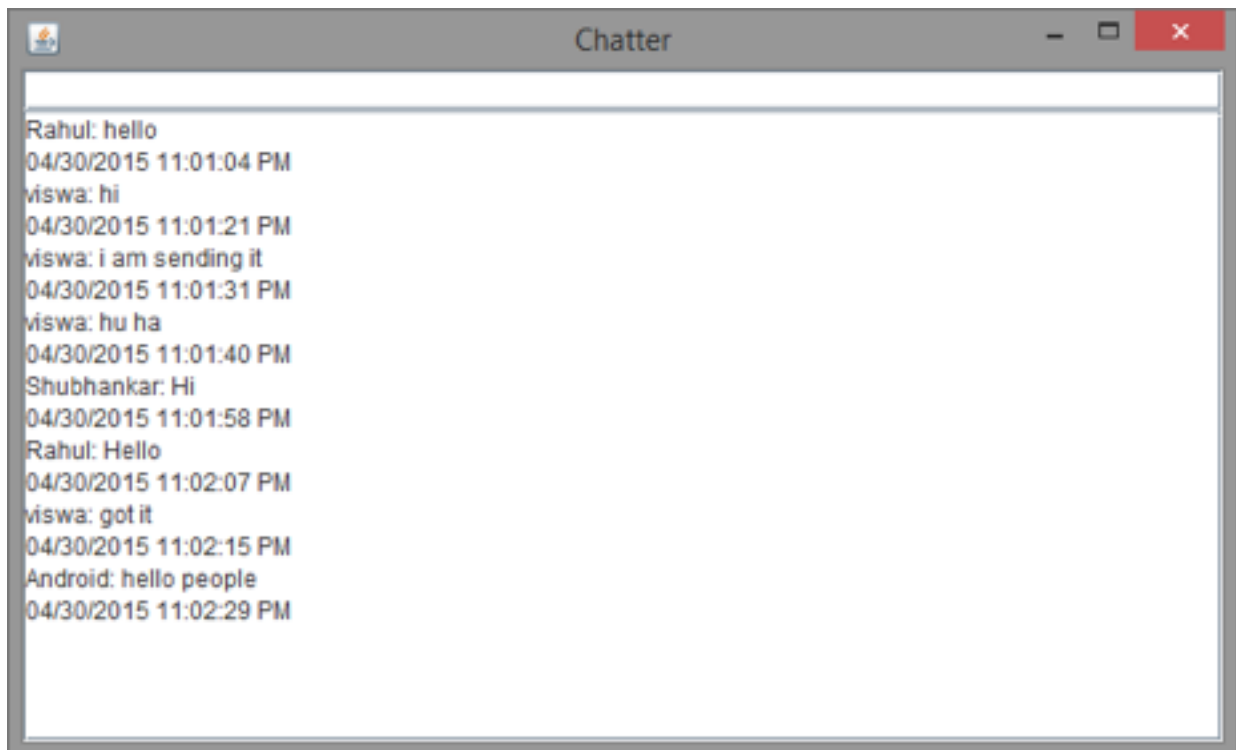


Fig 12 - Mixed Chat Session from Android and PC

Java Code :

Java Code for Chat Server:

```
package server;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashSet;

public class ChatServer {

    private static final int PORT = 9001;
    private static HashSet<String> names = new HashSet<String>();
    //hash table of all names
    private static HashSet<PrintWriter> messages = new HashSet<PrintWriter>();
    //hash table of all messages
    public static void main(String[] args) throws Exception {
        System.out.println("The chat server is running.");
        ServerSocket listener = new ServerSocket(PORT);
        try {
            while (true) {
                new Handler(listener.accept()).start();
            }
        } finally {
            listener.close();
        }
    }

    private static class Handler extends Thread {
        private String name;
        private Socket socket;
        private BufferedReader in;
        private PrintWriter out;
        public Handler(Socket socket) {
            this.socket = socket;
        }
        public void run() {
            try {

                // Create character streams for the socket.
                in = new BufferedReader(new InputStreamReader(
                    socket.getInputStream()));
                out = new PrintWriter(socket.getOutputStream(), true);
                while (true) {
                    out.println("SUBMITNAME");
                    name = in.readLine();
                    if (name == null) {
```

```
return;
    }
    synchronized (names) {
        if (!names.contains(name)) {
            names.add(name);
            break;
        }
    }
}

out.println("NAMEACCEPTED");
messages.add(out);
while (true) {
    String input = in.readLine();//read input from the stream
of the socket
    if (input == null) {
        return;
    }
    for (PrintWriterwriter : messages) {//send out the message to other users
with "message" appended at the beginning
writer.println("MESSAGE " + name + ": " + input);
    }
} catch (IOExceptione) {
System.out.println(e);
} finally {
if (name != null) {
names.remove(name);
}
if (out != null) {
messages.remove(out);
}
try {
socket.close();
} catch (IOExceptione) {
}
}
}
}}
```

Java Code for Client:

```
package lastchance;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.text.SimpleDateFormat;
```

```

import java.util.Date;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class last {

    BufferedReader in;
    PrintWriter out;
    JFrame frame = new JFrame("Chatter");
    JTextField textField = new JTextField(40);
    JTextArea messageArea = new JTextArea(20, 60);

    public last() {

        // Layout GUI
        textField.setEditable(false);
        messageArea.setEditable(false);
        frame.getContentPane().add(textField, "North");
        frame.getContentPane().add(new JScrollPane(messageArea), "Center");
        frame.pack();

        // Add Listeners
        textField.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                out.println(textField.getText());
                textField.setText("");
            }
        });

        private String getServerAddress() {
            return JOptionPane.showInputDialog(
                frame,
                "Enter IP Address of the Server:",
                "Welcome to the Chat application",
                JOptionPane.QUESTION_MESSAGE);
        }

        private String getName() {
            return JOptionPane.showInputDialog(
                frame,
                "enter your name:",
                "Screen name selection",
                JOptionPane.PLAIN_MESSAGE);
        }
    }
}

```

```

private void run() throws IOException {

    // open the socket
    String serverAddress = getServerAddress();
    Socket socket = new Socket(serverAddress, 9001);
    in = new BufferedReader(new InputStreamReader(
        socket.getInputStream()));
    out = new PrintWriter(socket.getOutputStream(), true);

    // Process all messages from server, according to the protocol.
    while (true) {
        String line = in.readLine();
        if (line.startsWith("SUBMITNAME")) {
            out.println(getName());
        } elseif (line.startsWith("NAMEACCEPTED")) {
            textField.setEditable(true);
        } elseif (line.startsWith("MESSAGE")) {
            messageArea.append(line.substring(8) + "\n");
            Date date = new Date();
            SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy h:mm:ss a");
            String formattedDate = sdf.format(date);
            System.out.println(formattedDate);
            messageArea.append(formattedDate + "\n");
            //time stamp
        }
    }
}

public static void main(String[] args) throws Exception {
    lastclient = new last();
    client.frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    client.frame.setVisible(true);
    client.run();
}
}

```

Code For Android App:

```

package com.example.waychat;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.io.*;
import java.net.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.view.View;
import android.widget.Button;

```

```

import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends Activity {

    int bytesRead=0;
    PrintWriter out;
    String nil=null;
    public void clearEnterName(){
        runOnUiThread(new Runnable() {
            public void run(){
                EditText text = (EditText) findViewById(R.id.entername);
                text.setText(nil);
            }
        });
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button send = (Button) findViewById(R.id.send);
        send.setOnClickListener(new View.OnClickListener() {
            public void onClick(View arg0) {
                EditText text = (EditText) findViewById(R.id.entername);
                String convText = text.getText().toString();
                out.println(convText);
                //text.setText(nil);
                clearEnterName();
            }
        });

        //String convText = text.getText().toString();

        Button start = (Button) findViewById(R.id.start);
        start.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Button button = (Button) v;
                button.setVisibility(View.GONE);
                Thread cThread = new Thread(new ClientThread());
                cThread.start();
            }
        });
        final class ClientThread implements Runnable {
            public void viewMessage(final String printThis) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        TextView text1 = (TextView)
                        findViewById(R.id.text1);
                        text1.append(printThis);
                        //EditText text8 =
                        (EditText) findViewById(R.id.entername);
                        //text8.setText(nil);

                        //stuff that updates ui
                    }
                });
            }
        }
    }
}

```

```

    }
});

}

    public void run() {

        EditText text7 =
        (EditText) findViewById(R.id. entername);
        String convtext = text7.getText().toString();
        clearentername();

        //text_vi text =
        (EditText) findViewById(R.id. entername);
        /*TextView text1 = (TextView)
        findViewById(R.id. text1);
        text1.setText(convtext);*/

        Socket sock1;

        try {
            sock1 = new Socket("ec2-54-68-62-42.us-
west-2.compute.amazonaws.com", 9001);
            viewmessage("Waiting...");

            viewmessage("Accepted connection : " + sock1);

            //counter++;
            // receive file

            // InputStream is = sock1.getInputStream();
            BufferedReader in =
            new BufferedReader(new InputStreamReader(sock1.getInputStream()));
            //BufferedReader br = new BufferedReader(i);
            //int bufsize = sock1.getReceiveBufferSize();
            //byte [] mybytearray = new byte [bufsize];
            //FileOutputStream fos = new
            FileOutputStream(Environment.getExternalStorageDirectory().getAbsolutePath()
            + "/Download/recieved1.jpg"); // destination path and name of file
            out = new PrintWriter(sock1.getOutputStream(), true);
            while (true) {
                String line = in.readLine();
                if (line.startsWith("SUBMITNAME")) {

                    viewmessage(convtext);
                    out.println(convtext);
                    clearentername();
                } elseif (line.startsWith("NAMEACCEPTED"))
                {
                    continue;
                } elseif (line.startsWith("MESSAGE")) {

```



```

        viewmessage(line.substring(8) + "\n");
        Date date = new Date();
        SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy
h:mm:ss a");
        String formattedDate =
sdf.format(date);
        System.out.println(formattedDate);
        viewmessage(formattedDate+"\n");
    }
}

//out.flush();

// out.close();

// sock1.close();
}

```

```

        catch (UnknownHostException) {
// TODO Auto-generated catch block
e.printStackTrace();
        } catch (IOException) {
// TODO Auto-generated catch block
e.printStackTrace();
        }

// TODO Auto-generated method stub
    }

    });
}

```

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

```

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {

    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}

```

```
    }  
}
```

XML File for the Android App:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:gravity="left"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context="com.example.waychat.MainActivity">  
    <ScrollView  
        android:layout_width="fill_parent"  
        android:layout_height="0dip"  
        android:layout_weight="1">  
    </ScrollView>  
  
    <TextView  
        android:id="@+id/text1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_above="@+id/entername"  
        android:layout_alignLeft="@+id/entername"  
        android:text="@string/hello_world"  
        android:textIsSelectable="true"  
        android:textStyle="bold"/>  
  
    <Button  
        android:id="@+id/start"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignBaseline="@+id/send"  
        android:layout_alignBottom="@+id/send"  
        android:layout_toRightOf="@+id/text1"  
        android:text="start"/>  
  
    <Button  
        android:id="@+id/send"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignLeft="@+id/entername"
```

```
android:layout_alignParentBottom="true"  
android:layout_marginBottom="33dp"  
android:text="send"/>
```

```
<EditText  
android:id="@+id/entername"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_above="@+id/send"  
android:layout_alignParentLeft="true"  
android:layout_marginLeft="18dp"  
android:ems="10"/>
```

```
</RelativeLayout>
```