SIKSHA 'O' ANUSANDHAN
DEEMED TO BE UNIVERSITY

Admission Batch: 2019-2020 **Session: 2020-2021**

End-Term Project Report

Computer Science Workshop 1 (CSE 2141)

Submitted by

Name: A S V K VINAYAK

Registration No.: 1941012434

Branch: CSE

Semester: 3rd Section: P

Department of Computer Science & Engineering

Faculty of Engineering & Technology (ITER)

Jagamohan Nagar, Jagamara, Bhubaneswar, Odisha - 751030

Contents

INTRODUCTION

Communication is a mean for people to exchange messages. It has started since the beginning of human creation. Distant communication began as early as 1800 century with the introduction of television, telegraph and then telephone. Interestingly enough, telephone communication stands out as the fastest growing technology, from fixed line to mobile wireless, from voice call to data transfer. The emergence of computer network and telecommunication technologies bears the same objective that is to allow people to communicate. All this while, much efforts has been drawn towards consolidating the device into one and therefore indiscriminate the services.

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 quit 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 start chatting with everyone (message is broadcasted to all connected users).

BATIYAO CHATAPP

PROBLEM STATEMENT

GUI based Chatting Application

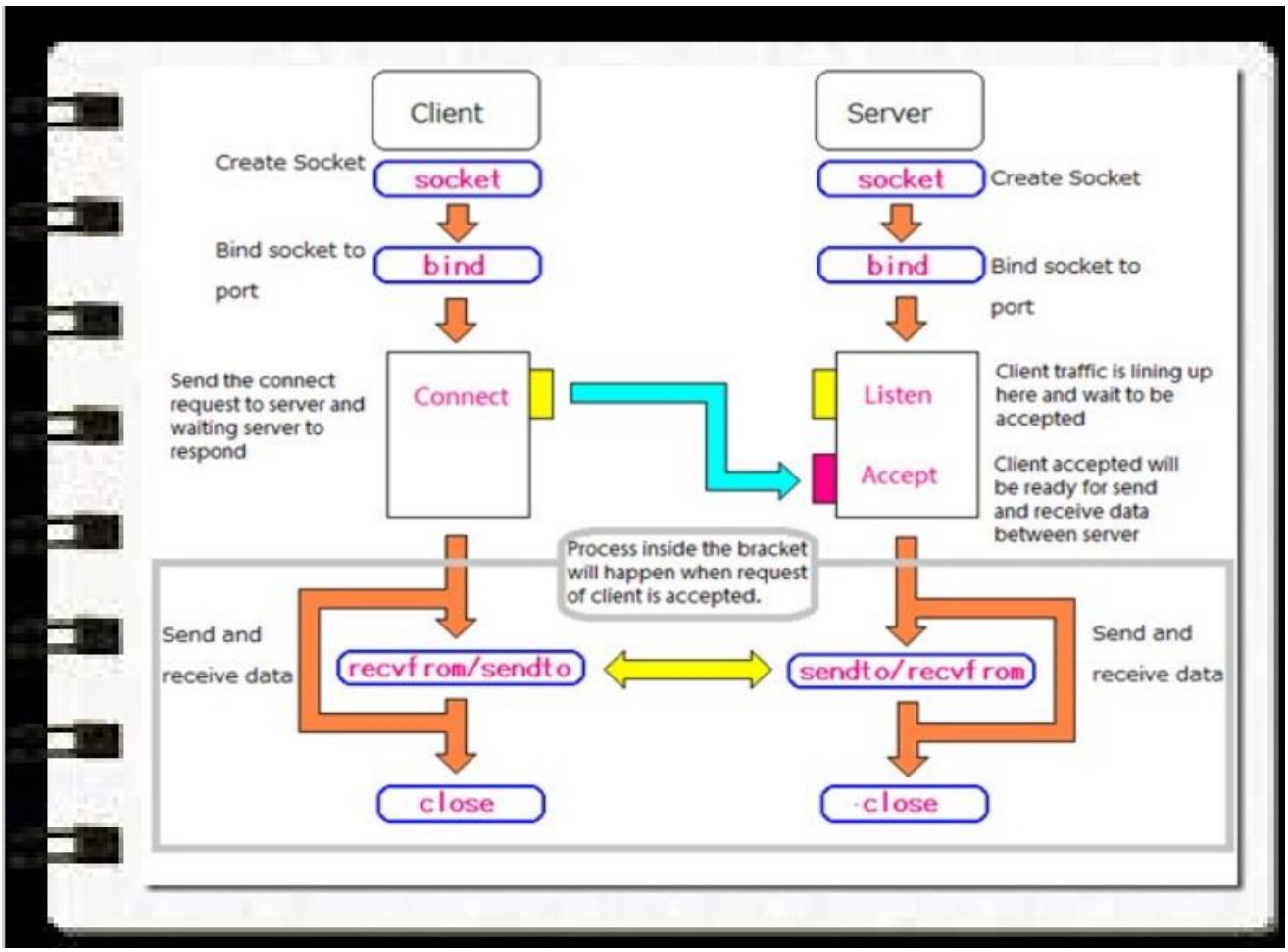
It is an implementation of multithreaded server. It accepts connections from an arbitrary number of clients; any message sent from one client is broadcast to all clients. In addition to ServerSockets, it demonstrates the use of threads. Because there are interactions among clients, this server needs to keep track of all the clients it has at any one time.

Instruction for implementation:

- Design GUI based inter face for client server communication.
- Server window should contain at least four buttons (START, STOP, RESPOND and ONLINE USERS). The start and stop button to start and stop the server, respond button to respond the client and online user will display the number of clients connected to the server.
- Client window should contain at least three buttons (CONNECT, DISCONNECT, SEND). Connect and disconnect to establish and close connection with server. By clicking the send button client will send the message. Client windows must display the client names like client1, client2 etc. The messages should be displayed in the text area.
- You can add more features and customize your chat application to make it more user friendly.

PROBEM ANALYSIS

- ❖We want to create an application by which we can have group conversations between us.
- ❖Each member will join a fixed port and they can all communicate with each other.
- ❖We will be creating a server though which all interactions are going to take place. The server is going to manage this using multithreading concept.
- ❖The server can also send responses to all its clients connected with it.
- ❖A client will send a message and that will be visible to every other member connected to the server including the client itself.



CODING PART (JAVA)

ChatClientCLI.java

- This Program is for thread purpose for client

```
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.HashMap;
import java.util.Scanner;

public class ChatClientCLI {
     private static HashMap<String, PrintWriter> connectedClients = new
HashMap<>();
     private static Socket clientSocket;
     private static ServerSocket server;
     private static BufferedReader in;
     private static class Listener implements Runnable {
          private BufferedReader in;
          @Override
          public void run() {
               try {
                    in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    String read;
                    for(;;) {
                         read = in.readLine();
                         if (read != null && !(read.isEmpty()))
System.out.println(read);
                    }
               } catch (IOException e) {
                    return;
               }
          }

     }
     private static class Writer implements Runnable {
          private PrintWriter out;
          private BufferedReader in;
          private String name="[SERVER]";
          @Override
          public void run() {
               Scanner write = new Scanner(System.in);
               try {
```

```
                    out = new
PrintWriter(clientSocket.getOutputStream(), true);

for(;;) {
                         if (write.hasNext())
out.println(write.nextLine());
                    }

               } catch (IOException e) {
                    write.close();
                    return;
               }
          }
     }

     public static void main(String[] args) {
          String ipAddress = null;
          if(args.length != 0) {
               ipAddress = args[0];
          } else {
               ipAddress = "localhost";
          }
          try {
               // this will try to create socket connection if server
socket exists
               clientSocket = new Socket(ipAddress, 4378);
          } catch (Exception e) {
               // throws an error if there is no server socket in that
port
               e.printStackTrace();
          }
          new Thread(new Writer()).start();
          new Thread(new Listener()).start();
     }
}
```

ChatServerCLI.java

- This Program is for thread purpose for server

```
import java.io.*;
import java.net.*;
import java.util.HashMap;

public class ChatServerCLI {
private static HashMap<String, PrintWriter> connectedClients = new
HashMap<>();
     private static final int MAX_CONNECTED = 50;
     private static int PORT;
     private static boolean verbose;
     private static ServerSocket server;

     // Start of Client Handler
     private static class ClientHandler implements Runnable {
          private Socket socket;
          private PrintWriter out;
          private BufferedReader in;
          private String name;

          public ClientHandler(Socket socket) {
               this.socket = socket;
          }

          @Override
          public void run(){
               if (verbose)
                    System.out.println("Client connected: " +
socket.getInetAddress());
               try {
                    in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
          out = new PrintWriter(socket.getOutputStream(), true);
                    for(;;) {
                         out.println("Enter username:\t");
                         name = in.readLine();
                         if (name == null) {
                              return;
                         }
                         synchronized (connectedClients) {
if (!name.isEmpty() && !connectedClients.keySet().contains(name))
break;
                              else out.println("INVALIDNAME");
                         }
```

```
                    }
out.println("Welcome to the chat group, " + name.toUpperCase() + "!");
                    if (verbose) System.out.println(name.toLowerCase() +
" has joined.");
broadcastMessage("[SYSTEM MESSAGE] " + name.toLowerCase() + " has
joined.");
                    connectedClients.put(name, out);
                    String message;
                    out.println("You may join the chat now...");
                    while ((message = in.readLine()) != null) {
                         if (!message.isEmpty()) {
               if (message.toLowerCase().equals("/quit")) break;
                              broadcastMessage(name + ": " + message);
                         }
                    }
               } catch (Exception e) {
                    if (verbose) System.out.println(e);
               } finally {
                    if (name != null) {
     if (verbose) System.out.println(" ["+name+"] " + " is leaving");
                         connectedClients.remove(name);
                         broadcastMessage(" ["+name+"]" +" has left");
                    }
               }
          }

     }
     // End of Client Handler

     private static void broadcastMessage(String message) {
          for (PrintWriter p: connectedClients.values()) {
               p.println(message);
          }
     }

     public static void start(boolean isVerbose) {
          verbose = isVerbose;
          try {
               server = new ServerSocket(getRandomPort());
               if (verbose) {
               System.out.println("Server started on port: " + PORT);
               System.out.println("Now listening for connections...");
               }
               for(;;) {
                    if (connectedClients.size() <= MAX_CONNECTED){
                         Thread newClient = new Thread(
                                   new ClientHandler(server.accept()));
                         newClient.start();
                    }
```

```
            }
        }
        catch (Exception e) {
            if (verbose) {
                System.out.println("\nError occured: \n");
                e.printStackTrace();
                System.out.println("\nExiting...");
            }
        }
    }

    public static void stop() throws IOException {
        if (!server.isClosed()) server.close();
    }

    private static int getRandomPort() {
        int port = FreePortFinder.findFreeLocalPort();
        PORT = port;
        return port;
    }

    public static void main(String[] args) throws IOException {
  start(args[0].toLowerCase().equals("verbose") ? true : false);
    }
}
```

TextAreaOutputStream.java
This Program is to take input from clients

```
import java.io.IOException;
import java.io.OutputStream;
import javax.swing.JTextArea;
public class TextAreaOutputStream extends OutputStream {
    private final JTextArea txtArea;
    private final StringBuilder sb = new StringBuilder();
    public TextAreaOutputStream(JTextArea txtArea) {
        this.txtArea = txtArea;
    }
    @Override
    public void write(int b) throws IOException {
        if (b == '\r')
        {          return;         }
       if (b == '\n') {
           final String text = sb.toString() + "\n";
           txtArea.append(text);
           sb.setLength(0);
       } else {
           sb.append((char) b);
       }
```

```
    }
}
```

FreePortFinder.java
This Program is to get random port numbers to connect server and clients.

```
import java.io.IOException;
import java.net.ServerSocket;
public final class FreePortFinder {

    private FreePortFinder() {
    }

    public static int findFreeLocalPort() {
        try {
            ServerSocket serverSocket = new ServerSocket(0);
            int port = serverSocket.getLocalPort();
            serverSocket.close();
            return port;
        } catch (IOException e) {
            throw new IllegalStateException(e);
        }
    }
}
```

ClientUI.java
This Program is the main chat system for client side, with GUI.

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

public class ClientUI extends JFrame implements ActionListener {

    private static final long serialVersionUID = 1L;
    // Socket Related
    private static Socket clientSocket;
    private static int PORT;
    private PrintWriter out;

    // JFrame related
    private JPanel contentPane;
    private JTextArea txtAreaLogs;
```

```
    private JButton btnStart;
    private JPanel panelNorth;
    private JLabel lblChatClient;
    private JPanel panelNorthSouth;
    private JLabel lblPort;
    private JLabel lblName;
    private JPanel panelSouth;
    private JButton btnSend;
    private JTextField txtMessage;
    private JTextField txtNickname;
    private JTextField txtPort;
    private String clientName;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    ClientUI frame = new ClientUI();

    UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsL
ookAndFeel");
                    SwingUtilities.updateComponentTreeUI(frame);
                    //Logs
                    System.setOut(new PrintStream(new
TextAreaOutputStream(frame.txtAreaLogs)));
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public ClientUI() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 570, 400);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(new BorderLayout(0, 0));

        panelNorth = new JPanel();
        contentPane.add(panelNorth, BorderLayout.NORTH);
```

```
        panelNorth.setLayout(new BorderLayout(0, 0));

        lblChatClient = new JLabel("CHAT CLIENT");
        lblChatClient.setHorizontalAlignment(SwingConstants.CENTER);
        lblChatClient.setFont(new Font("Tahoma", Font.PLAIN, 40));
        panelNorth.add(lblChatClient, BorderLayout.NORTH);

        panelNorthSouth = new JPanel();
        panelNorth.add(panelNorthSouth, BorderLayout.SOUTH);
    panelNorthSouth.setLayout(new FlowLayout(FlowLayout.RIGHT, 5, 5));

        lblName = new JLabel("Nickname");
        lblName.setFont(new Font("Arial", Font.PLAIN, 18));
        panelNorthSouth.add(lblName);

        txtNickname = new JTextField();
        txtNickname.setColumns(15);
        panelNorthSouth.add(txtNickname);

        lblPort = new JLabel("Port");
        lblPort.setFont(new Font("Arial", Font.PLAIN, 18));
        panelNorthSouth.add(lblPort);

        txtPort = new JTextField();
        panelNorthSouth.add(txtPort);
        txtPort.setColumns(15);

        btnStart = new JButton("CONNECT");
        panelNorthSouth.add(btnStart);
        btnStart.addActionListener(this);
        btnStart.setFont(new Font("Tahoma", Font.PLAIN, 12));

        JScrollPane scrollPane = new JScrollPane();
        contentPane.add(scrollPane, BorderLayout.CENTER);

        txtAreaLogs = new JTextArea();
        txtAreaLogs.setBackground(Color.BLACK);
        txtAreaLogs.setForeground(Color.WHITE);
        txtAreaLogs.setLineWrap(true);
        scrollPane.setViewportView(txtAreaLogs);

        panelSouth = new JPanel();
        FlowLayout fl_panelSouth = (FlowLayout)
panelSouth.getLayout();
        fl_panelSouth.setAlignment(FlowLayout.RIGHT);
        contentPane.add(panelSouth, BorderLayout.SOUTH);

        txtMessage = new JTextField();
        panelSouth.add(txtMessage);
```

```
        txtMessage.setColumns(50);

        btnSend = new JButton("SEND");
        btnSend.addActionListener(this);
        btnSend.setFont(new Font("Tahoma", Font.PLAIN, 18));
        panelSouth.add(btnSend);
    }

    @Override
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource() == btnStart) {
            if(btnStart.getText().equals("CONNECT")) {
                btnStart.setText("DISCONNECT");
                start();
            }else {
                btnStart.setText("CONNECT");
                stop();
            }
        }else if(e.getSource() == btnSend) {
            String message = txtMessage.getText().trim();
            if(!message.isEmpty()) {
                out.println(message);
                txtMessage.setText("");

            }
        }
        //Refresh UI
        refreshUIComponents();
    }

    public void refreshUIComponents()
    {
        return;
    }

    public void start() {
        try {
            PORT = Integer.parseInt(txtPort.getText().trim());
            clientName = txtNickname.getText().trim();
            clientSocket = new Socket("localhost", PORT);
        out = new PrintWriter(clientSocket.getOutputStream(), true);
            new Thread(new Listener()).start();
            //send name
            out.println(clientName);
        } catch (Exception err) {
            addToLogs("[ERROR] "+err.getLocalizedMessage());
        }
    }
```

```
    public void stop(){
        if(!clientSocket.isClosed()) {
            try {
                clientSocket.close();
            } catch (IOException e1) {}
        }
    }

    public static void addToLogs(String message) {
        System.out.printf("%s %s\n", ServerUI.formatter.format(new
Date()), message);
    }

    private static class Listener implements Runnable {
        private BufferedReader in;
        @Override
        public void run() {
            try {
                in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                String read;
                for(;;) {
                    read = in.readLine();
                    if (read != null && !(read.isEmpty()))
addToLogs(read);
                }
            } catch (IOException e) {
                return;
            }
        }

    }
}
```

ServerUI.java
This Program is the main chat system for server side, with GUI.

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import java.text.*;
import java.util.*;
import java.util.Map.Entry;

public class ServerUI extends JFrame implements ActionListener {

     private static final long serialVersionUID = 1L;
     // Socket Related

     public static SimpleDateFormat formatter = new
SimpleDateFormat("[hh:mm a]");
     private static HashMap<String, PrintWriter> connectedClients = new
HashMap<>();
     private static final int MAX_CONNECTED = 50;
     private static int PORT;
     private static ServerSocket server;
     private static volatile boolean exit = false;
     // JFrame related
     private JPanel contentPane;
     private JTextArea txtAreaLogs;
     private JButton btnStart;
     private JLabel lblChatServer;
     private Container panelSouth;

     /**
      * Launch the application.
      */
     public static void main(String[] args) {
          EventQueue.invokeLater(new Runnable() {
               public void run() {
                    try {
                         ServerUI frame = new ServerUI();
     UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsL
ookAndFeel");
                         SwingUtilities.updateComponentTreeUI(frame);
                         //Logs
```

```
                        System.setOut(new PrintStream(new
TextAreaOutputStream(frame.txtAreaLogs)));
                        frame.setVisible(true);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
        }

        /**
         * Create the frame.
         */
        public ServerUI()
        {
            super("Batiyao ChatApp");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(400, 400, 600, 400);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            contentPane.setLayout(new BorderLayout(0, 0));

            lblChatServer = new JLabel("CHAT SERVER");
            lblChatServer.setHorizontalAlignment(SwingConstants.CENTER);
            lblChatServer.setFont(new Font("Tahoma", Font.PLAIN, 40));
            contentPane.add(lblChatServer, BorderLayout.NORTH);

            btnStart = new JButton("START");
            btnStart.addActionListener(this);
            btnStart.setFont(new Font("Tahoma", Font.PLAIN, 30));
            contentPane.add(btnStart, BorderLayout.WEST);

            JScrollPane scrollPane = new JScrollPane();
            contentPane.add(scrollPane, BorderLayout.CENTER);

            txtAreaLogs = new JTextArea();
            txtAreaLogs.setBackground(Color.BLACK);
            txtAreaLogs.setForeground(Color.WHITE);
            txtAreaLogs.setLineWrap(true);
            scrollPane.setViewportView(txtAreaLogs);


            panelSouth = new JPanel();
     FlowLayout fl_panelSouth = (FlowLayout) panelSouth.getLayout();
            fl_panelSouth.setAlignment(FlowLayout.RIGHT);
            contentPane.add(panelSouth, BorderLayout.SOUTH);

            JLabel namelabel = new JLabel("Made By Vinayak");
            namelabel.setForeground(Color.BLACK);
```

```
          namelabel.setFont(new Font("Arial", Font.PLAIN, 25));
          namelabel.setBackground(Color.DARK_GRAY);
          panelSouth.add(namelabel,BorderLayout.WEST);

          JButton btnRespond = new JButton("RESPOND");
          btnRespond.addActionListener(new ActionListener() {
               @SuppressWarnings("unused")
               public void actionPerformed(ActionEvent e) {
                    if(false) {
JOptionPane.showMessageDialog(null,"Server not started yet!");
                    }else if(connectedClients.size()==0) {
JOptionPane.showMessageDialog(null,"No clients Available now!");
                         return;
                    }
String msgString=JOptionPane.showInputDialog("Enter your response to
all users!");
                    if(msgString==null) return;
                    broadcastMessage(" [SERVER] : "+msgString);
               }
          });
          btnRespond.setForeground(Color.WHITE);
          btnRespond.setFont(new Font("Arial", Font.PLAIN, 25));
          btnRespond.setBorderPainted(false);
          btnRespond.setBackground(Color.DARK_GRAY);
          btnRespond.setBounds(100, 393, 150, 80);
          panelSouth.add(btnRespond);

          JButton btnclient = new JButton("ONLINE USERS");
          btnclient.setForeground(Color.WHITE);
          btnclient.setFont(new Font("Arial", Font.PLAIN, 25));
          btnclient.setBorderPainted(false);
          btnclient.setBackground(Color.DARK_GRAY);
          btnclient.setBounds(100, 393, 150, 80);
          panelSouth.add(btnclient);

          btnclient.addActionListener(new ActionListener() {
               @SuppressWarnings({ "unused", "rawtypes" })
               public void actionPerformed(ActionEvent e) {
                     Set set = connectedClients.entrySet();
                    // Get an iterator
                    Iterator i = set.iterator();
                    // Display elements
               System.out.println("Current Online Users are
"+connectedClients.size());
                         int c=1;
                         while(i.hasNext()) {
                              Map.Entry me = (Entry) i.next();
                              System.out.println(c+"  [ "+me.getKey()+" ]
");
```

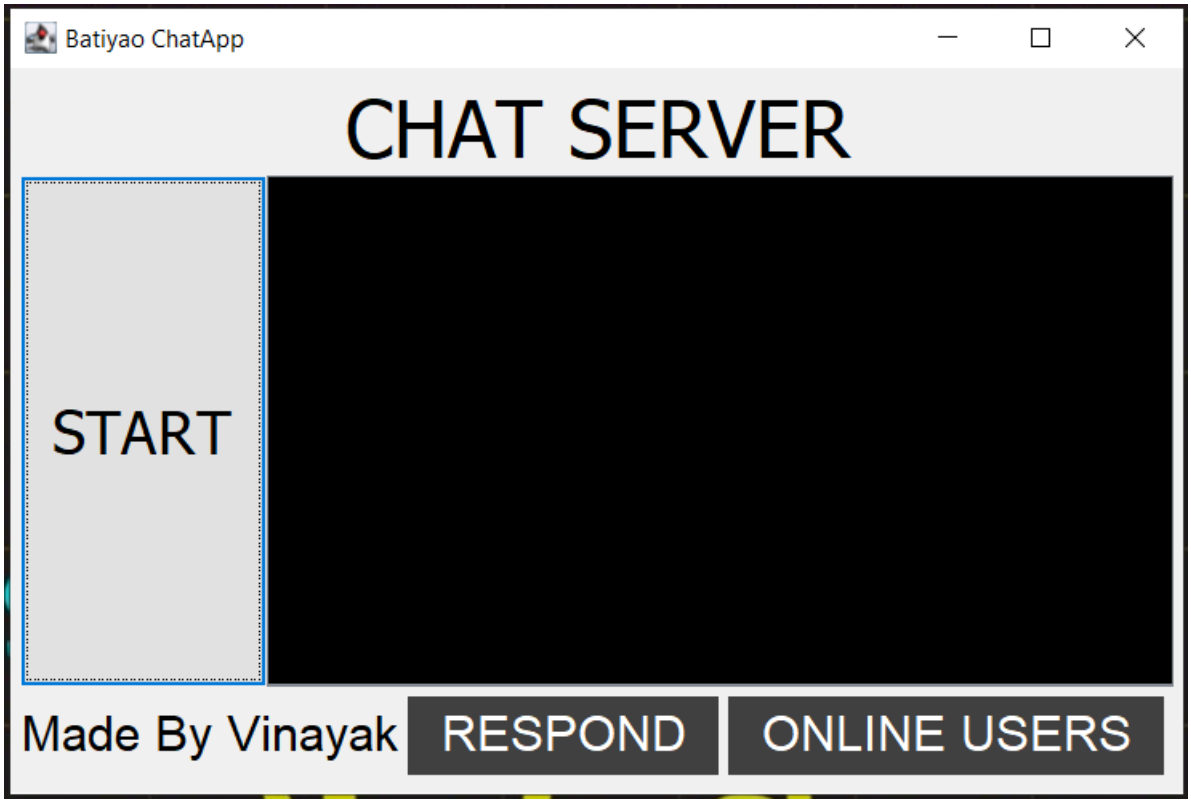
```
                    c++;
                }
            }
        });
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == btnStart) {
            if(btnStart.getText().equals("START")) {
                exit = false;
                getRandomPort();
                start();
                btnStart.setText("STOP");
            }else {
                addToLogs("Chat server stopped...");
                exit = true;
                btnStart.setText("START");
            }
        }

        //Refresh UI
        refreshUIComponents();
    }

    public void refreshUIComponents() {
        lblChatServer.setText("CHAT SERVER" + (!exit ? ":
"+PORT:""));
    }

    public static void start() {
        new Thread(new ServerHandler()).start();
    }

    public static void stop() throws IOException {
        if (!server.isClosed()) server.close();
    }

    private static void broadcastMessage(String message) {
        for (PrintWriter p: connectedClients.values()) {
            p.println(message);
        }
    }

    public static void addToLogs(String message) {
        System.out.printf("%s %s\n", formatter.format(new Date()),
message);
    }
```

```
    private static int getRandomPort() {
        int port = FreePortFinder.findFreeLocalPort();
        PORT = port;
        return port;
    }

    private static class ServerHandler implements Runnable{
        @Override
        public void run() {
            try {
                server = new ServerSocket(PORT);
                addToLogs("Server started on port: " + PORT);
                addToLogs("Now listening for connections...");
                while(!exit) {
                    if (connectedClients.size() <= MAX_CONNECTED){
            new Thread(new ClientHandler(server.accept())).start();
                    }
                }
            }
            catch (Exception e) {
                addToLogs("\nError occured: \n");
                addToLogs(Arrays.toString(e.getStackTrace()));
                addToLogs("\nExiting...");
            }
        }
    }

    // Start of Client Handler
    private static class ClientHandler implements Runnable {
        private Socket socket;
        private PrintWriter out;
        private BufferedReader in;
        private String name;

        public ClientHandler(Socket socket) {
            this.socket = socket;
        }

        @Override
        public void run(){
            addToLogs("Client connected: " +
socket.getInetAddress());
            try {
                in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(), true);
                for(;;) {
                    name = in.readLine();
                    if (name == null) {
```

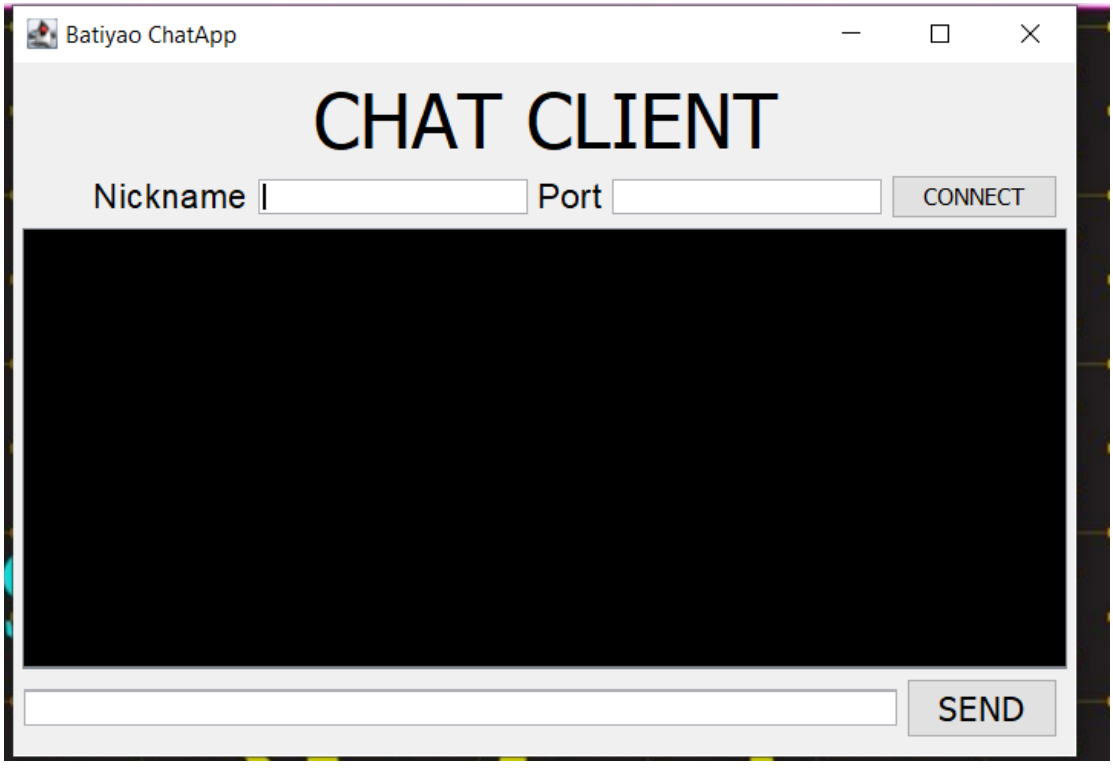
```
                    return;
                }
                synchronized (connectedClients) {
                    if (!name.isEmpty() &&
!connectedClients.keySet().contains(name)) break;
                    else out.println("INVALIDNAME");
                }
            }
            out.println("Welcome to the chat group, " +
name.toUpperCase() + "!");
            addToLogs(name.toUpperCase() + " has joined.");
broadcastMessage("[SYSTEM] " + name.toUpperCase() + " has joined.");
            connectedClients.put(name, out);
            String message;
            out.println("You may join the chat now...");
            while ((message = in.readLine()) != null && !exit) {
                if (!message.isEmpty()) {
                if (message.toLowerCase().equals("/quit")) break;
        broadcastMessage(String.format("[%s] %s",name, message));
                }
            }
        } catch (Exception e) {
            addToLogs(e.getMessage());
        } finally {
            if (name != null) {
                addToLogs(" ["+name+"] " + " is leaving");
                connectedClients.remove(name);
                broadcastMessage(" ["+name+"] "+ " has left");
            }
        }
    }

}
```

Snapshot of input and output

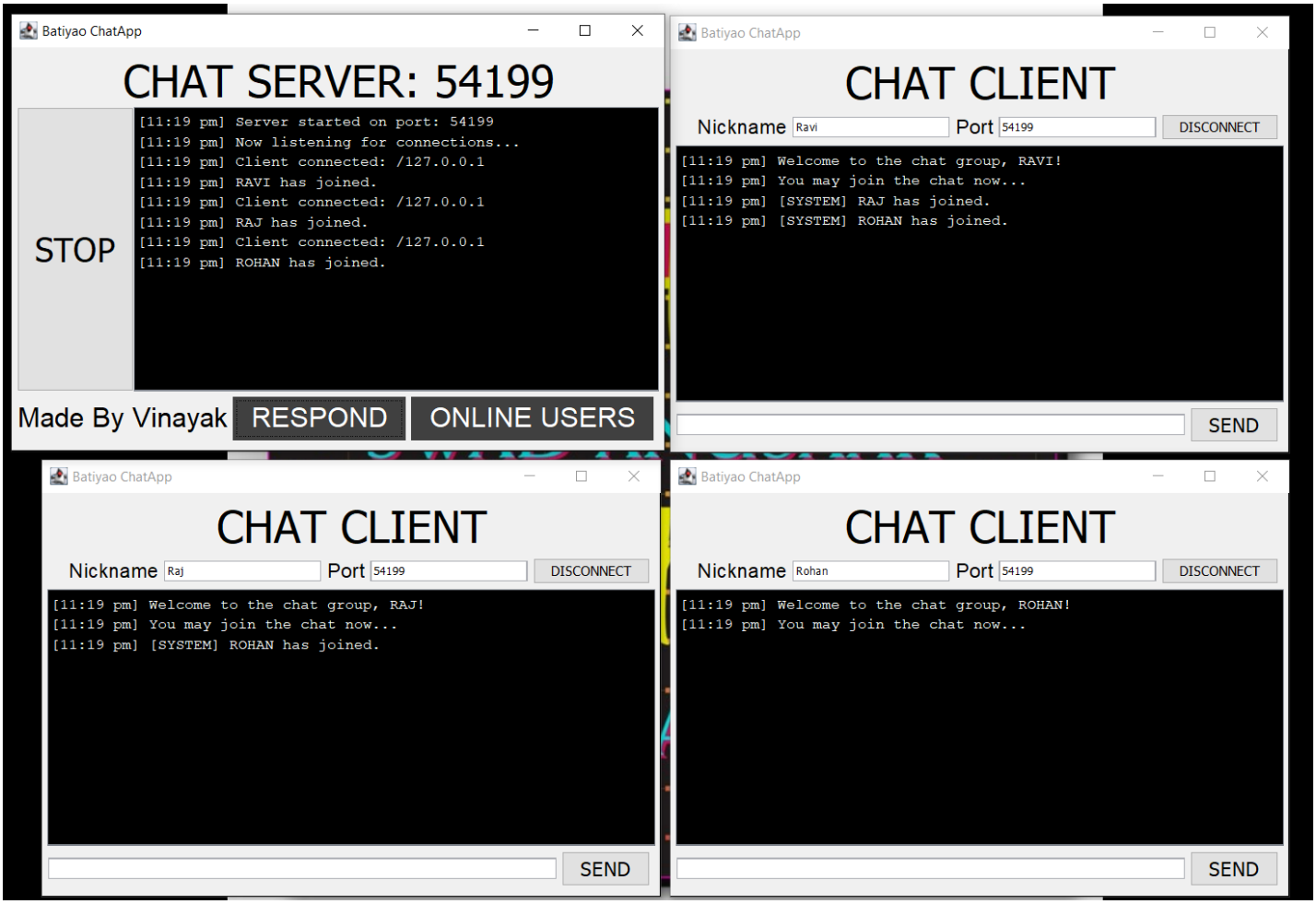
- Server Window



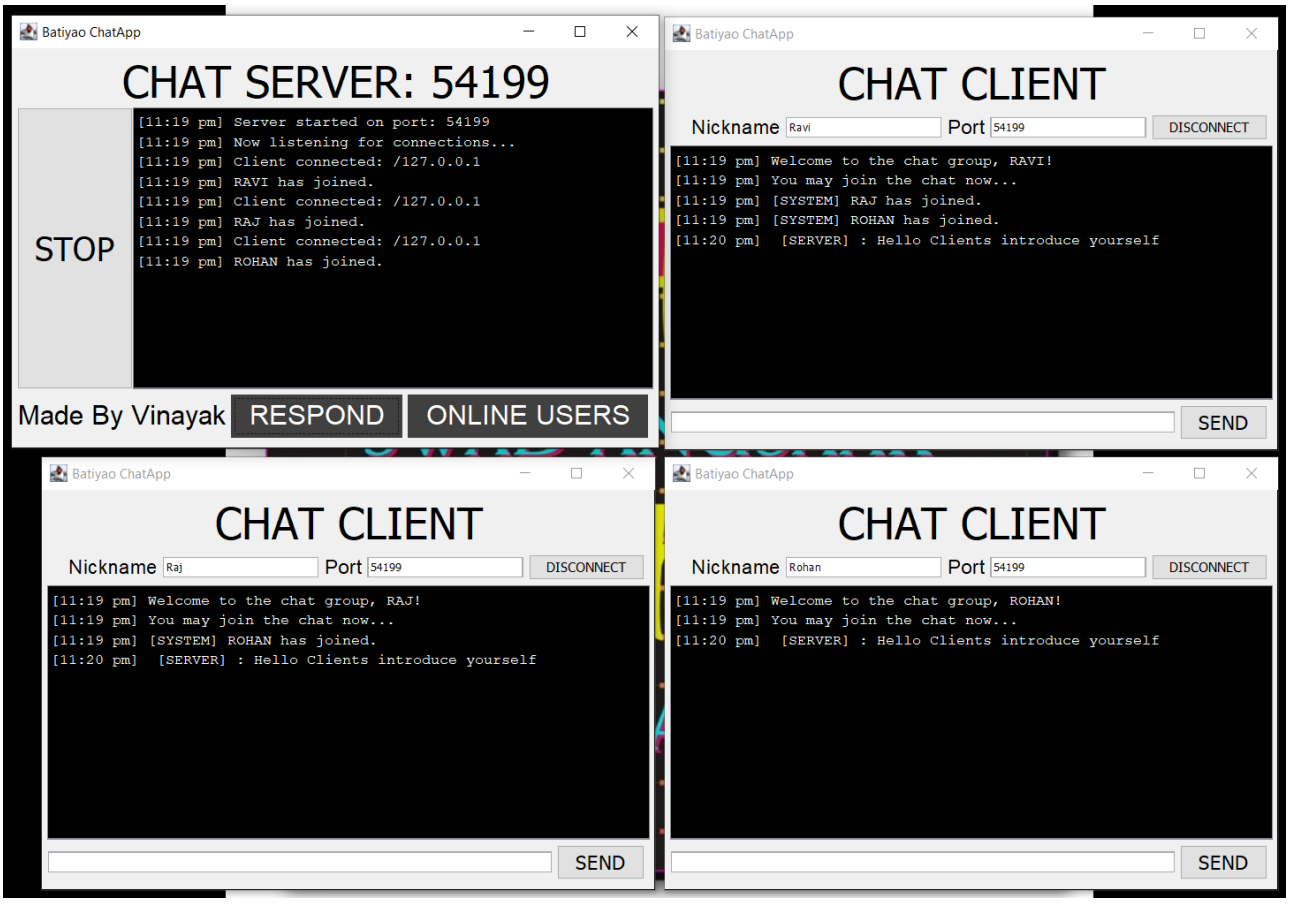
- Client Window



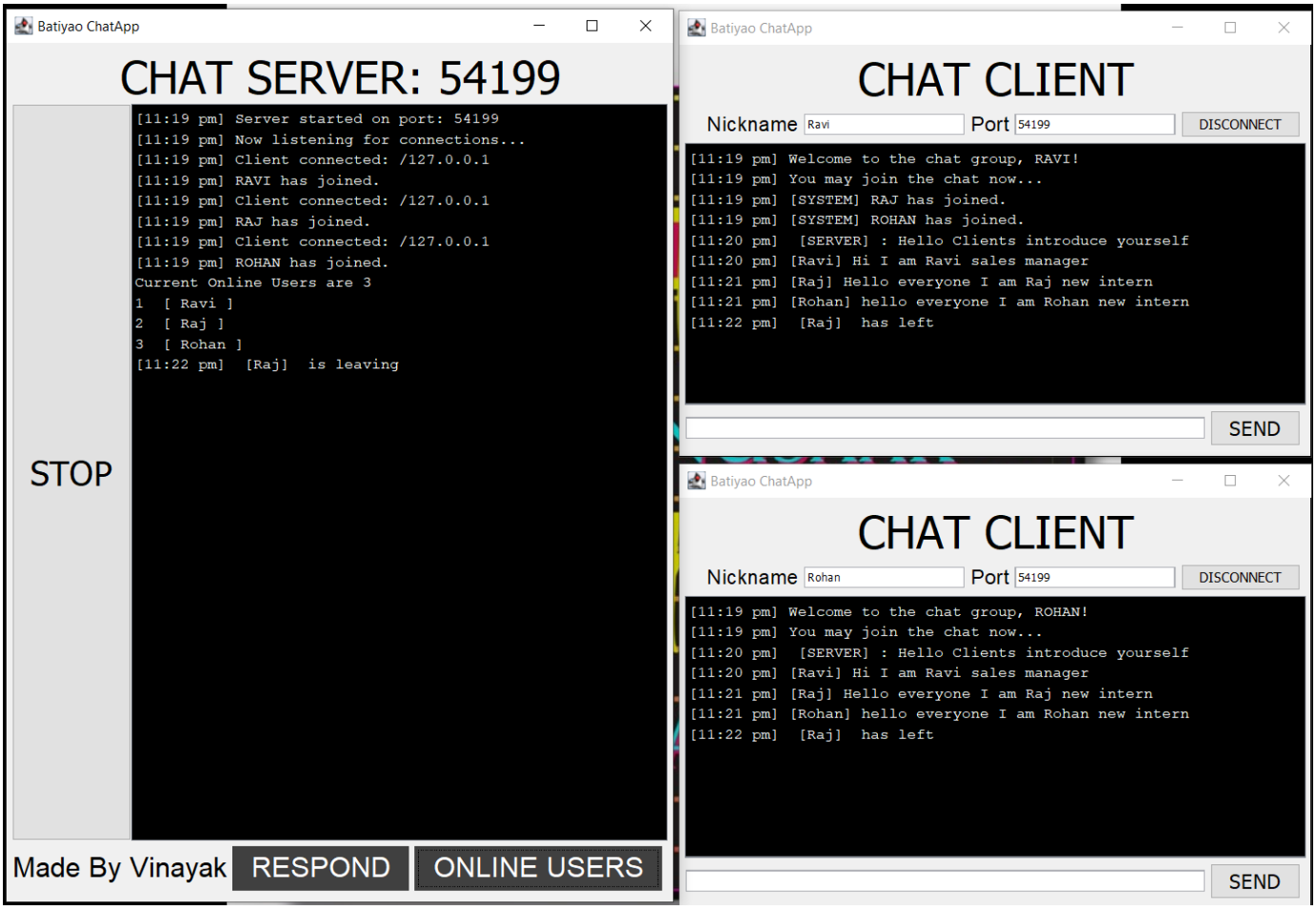
- Server and Client Connected



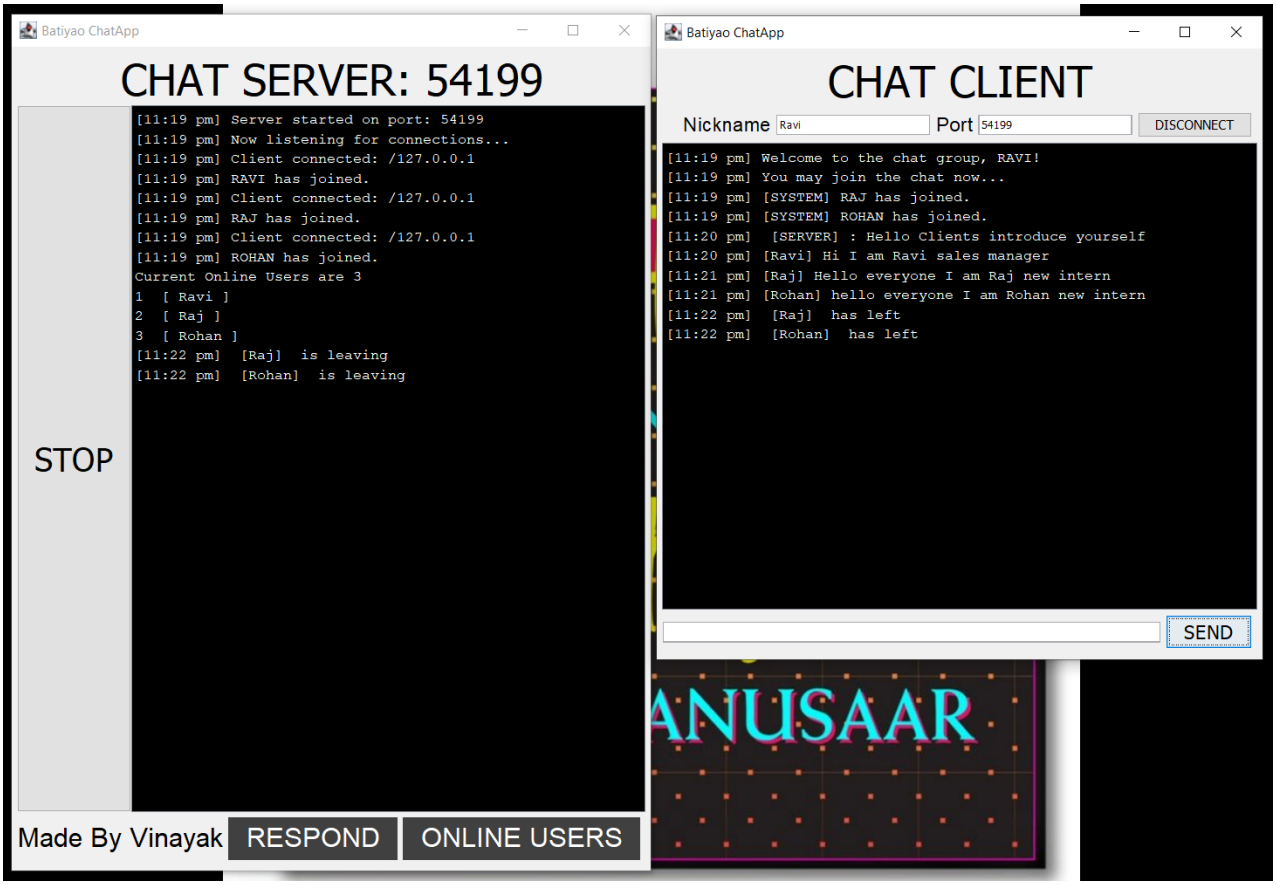
- Server sending responses



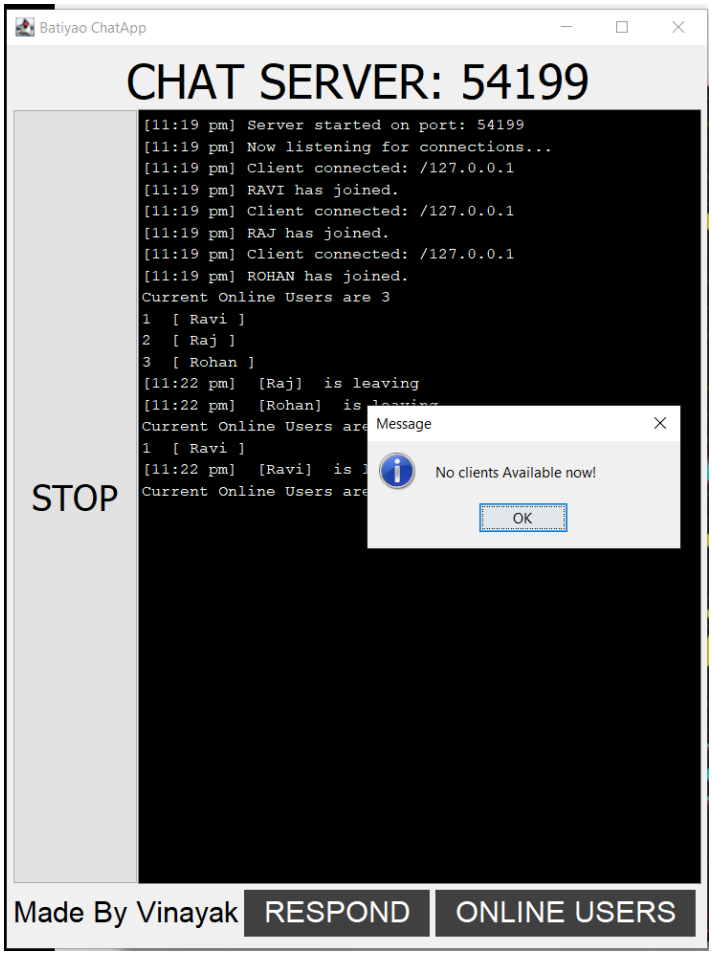
- Clients were chatting with each others and after sometime they left the chat.



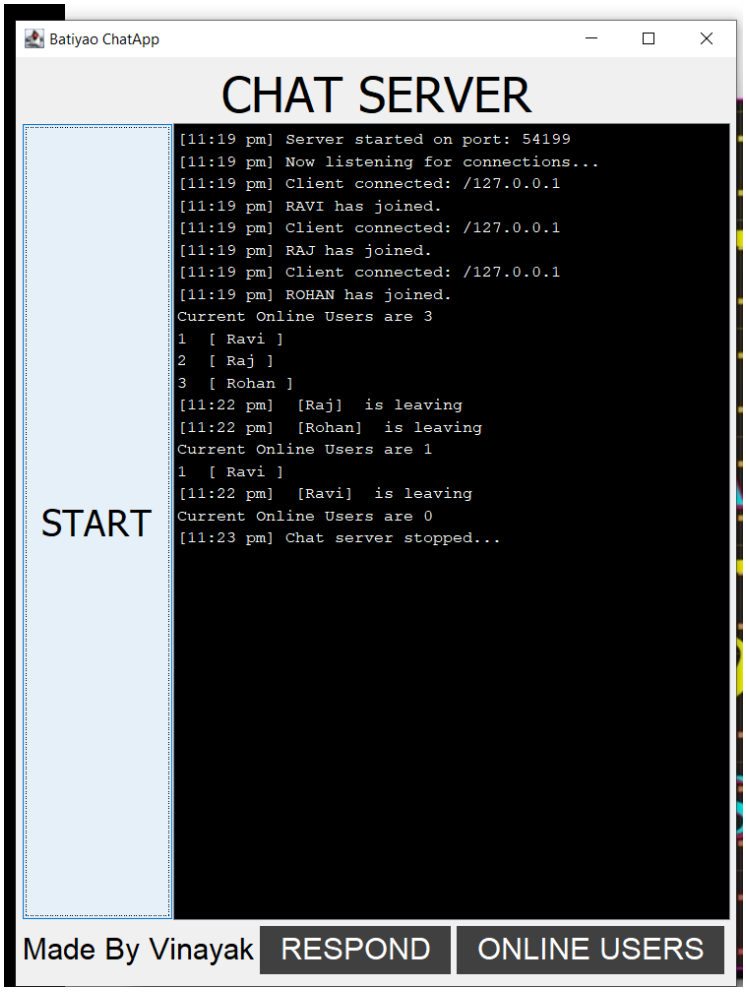
- We can see which clients left the chat from server



- Server can see which clients are online.



- Server is stopped.



Conclusion

The primary goal of this project is to give an idea about Client Server Chat. This project has given us an in depth information about java networking and its applications in day today life. If internet lives up to its potential, it will revolutionize the way people interact with information technology.

Author: A S V K VINAYAK

Date: 18/02/2021

THANK YOU