CSE3011 NETWORK PROGRAMMING

Mini Project Report

NAME- B.PRATYUSH

REGISTRATION NUMBER – 19BCN7114

LAB SLOT L1+L2 L11+L12 L43+L44

VIT-AP UNIVERSITY

GUIDED BY PROF. MUNEESWARI

TABLE OF CONTENTS

S.NO	INDEX	PAGE NO.
1	TITLE	3
2	ABSTRACT	3
3	PROJECT DESCRIPTION	3
4	MODULES DESIGN	4
5	SYSTEM DIAGRAM	5
6	SOURCE CODES	6
7	OUTPUTS	29

TITLE

VChatt! – A multi chat application server developed using Java

ABSTRACT

The project depicts the usage of Java as the programming language to build a chat server which can enable clients to communicate with each other by connecting to the server via sockets. The aim of this project is to make multi users communicate privately as well as broadcast a public message and avail other features of the chat session.

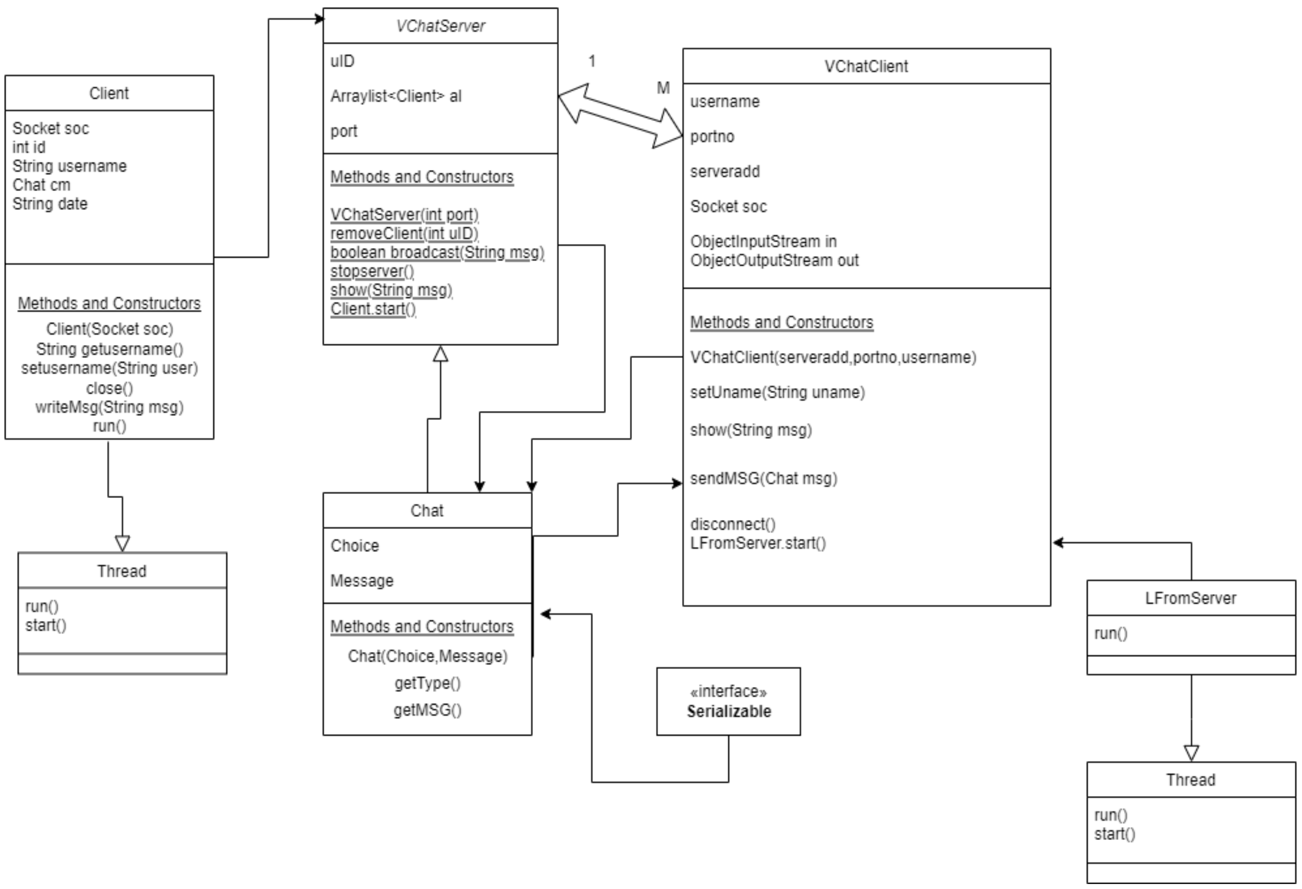
PROJECT DESCRIPTION

The server listens to requests from clients and stores a log of events happening in the application. The server can be started on any available ports. The client has to connect to the server via sockets to establish a connection. Server socket is used on the server end which accepts the client socket created on client end.

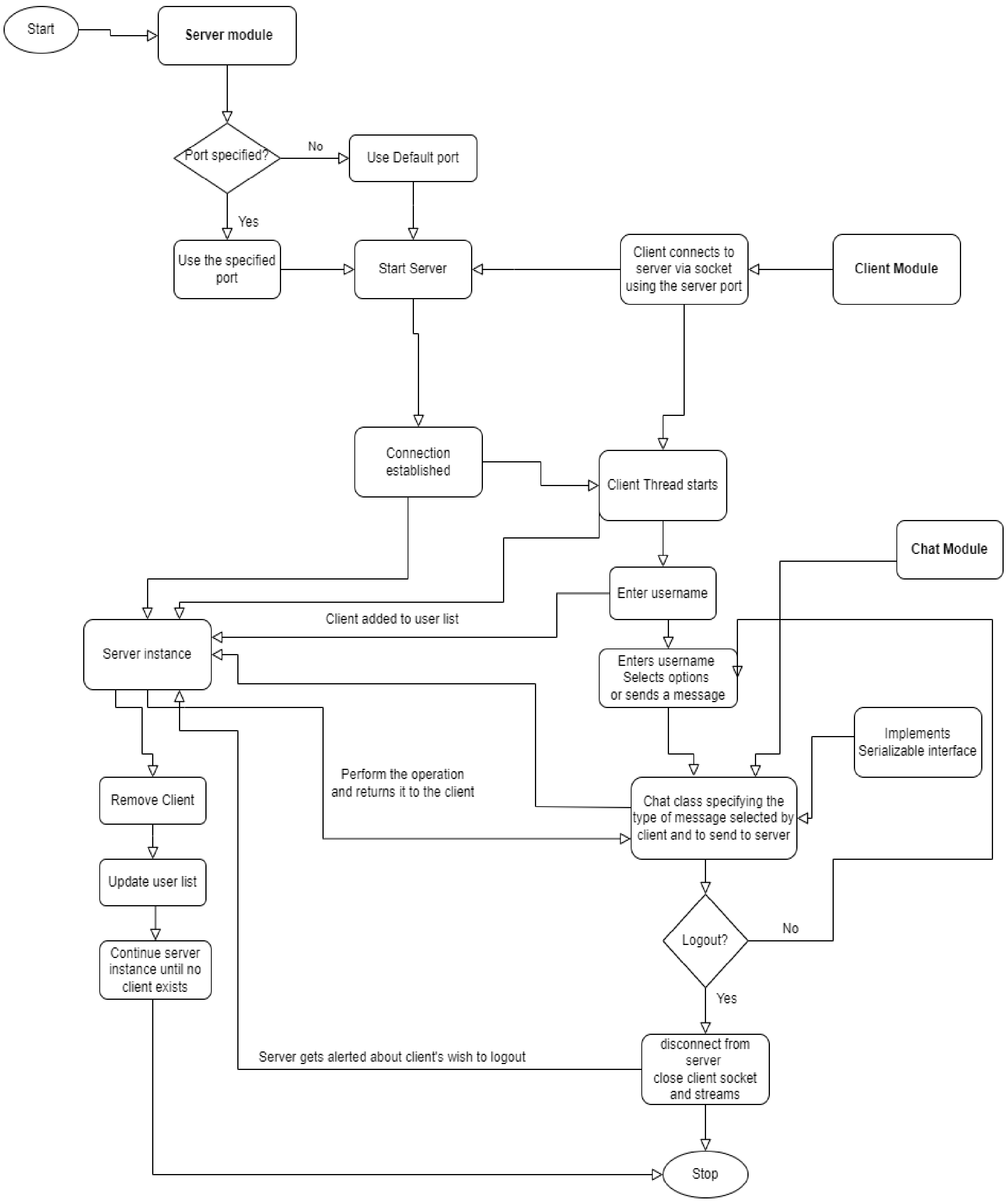
They are bound using the port no as the common entity. Once the connection is established, the client has to set up his/her username for the session. Each client runs on a separate thread and executes the required options provided by the server. Here we are using the concept of multithreading to split up the clients with separate threads to perform parallel activity simultaneously.

Client connects to server, sets up his/her username. Once the username is set up, server enters the username into the log and client thread is run for the specific client and a timestamp is assigned to the client. Few set of options are enabled on the client which are Sending a private message, sending a broadcast message, Finding the list of users in the session, finding the no of users active and logout option. For sending a broadcast message the client can directly send a message. For sending a private message the client has to enter the message in the following format: '@username message' so that a private message can be sent to the targeted username. To find the names of the users in the session, the client has to type 'WHOISIN' in CLI to get the list of users present and type 'U' to find the count of users active in the session. To log out of the session, the client has to type 'LOGOUT'. The server updates the user list and the server log. The transfer of data is done with the help of input and output streams. When users disconnect, the input and output streams are flushed and the socket is closed for the specific client.

MODULES DESIGN



SYSTEM DIAGRAM



Start
Server module
Port specified?
No
Use Default port
Yes
Use the specified port
Start Server
Client connects to server via socket using the server port
Client Module
Connection established
Client Thread starts
Chat Module
Enter username
Client added to user list
Server instance
Enters username Selects options or sends a message
Implements Serializable interface
Perform the operation and returns it to the client
Chat class specifying the type of message selected by client and to send to server
Remove Client
Update user list
Continue server instance until no client exists
Logout?
No
Yes
disconnect from server close client socket and streams
Server gets alerted about client's wish to logout
Stop

SOURCE CODES

Chat.java

```
import java.io.*;

//Class which sets the option type and sets the message to be passed

//It consists the option type and sets message

//Serializable interface is implemented to make it easier to access objects and variables and it makes
storing

//and sending objects easy

public class Chat implements Serializable{

  static final int WHOISIN =0, MSG=1, UserCount=2, Logout=3;

  //WHOISIN - shows the active users in the chat room

  // MSG- is used whenever we intend to send a message

  // UserCount- Prints the number of users connected to server.

  //Logout- disconnects user from chat when selected

  private int type;

  private String msg;


  Chat(int type,String msg)

  {

    this.type=type;

    this.msg=msg;

  }


  int getType()

  {

    return type;
  }


  String getMSG()
  {
    return msg;
  }
}
```

VChatClient.java

```
import java.net.*;
import java.util.*;
import java.io.*;
//Client side program
public class VChatClient{


   private String not= "***!!";
   //Making using of Object Input and Output streams to transfer data through sockets.
   private ObjectInputStream in;
   private ObjectOutputStream out;
   //Client Socket
   private Socket soc;
   // Server address, username and port variables
   private String server,uname;
   private int portno;
   //Client constructor
   VChatClient(String server,int portno,String uname)
   {
       this.server=server;
```

```
    this.portno=portno;

    this.uname=uname;

 }

//Setter and getter  Method to set and retrieve username

 public void setUname(String uname)

 {

    this.uname=uname;

 }

 public String getUname()

 {

    return uname;

 }

 //A display method to display strings

 private void show(String Msg)

 {

    System.out.println(Msg);

 }

  //Sends Message to server

 void sendMsg(Chat msg) {

   try

   {

      out.writeObject(msg);

   }

   catch(IOException e)

   {

      show("Exception arised while writing to server: " + e);

   }
```

```
}
 //A method to close Socket and the streams when user logs out or any issue arises.
 private void disconnect() {
 try
  {
     if(in != null)
     {
         in.close();
     }


  }
  catch(Exception e) {}
  try
  {
     if(out != null)
     {
         out.close();
     }
  }
  catch(Exception e) {}
  try
  {
     if(soc != null)
     {
         soc.close();
     }
```

```
      }
     catch(Exception e) {}


    }


    //the start() method starts the chat service
    public boolean start(){
       try
       {
          //Establishing client socket by passing server add and port no
          soc=new Socket(server,portno);
          System.out.println("Client Connection");
       }
       catch (Exception e) {
          //TODO: handle exception
          show("OOPS!Facing issues connecting to server: "+e);
          return false;
       }
        String Message = "Connection established successfully " + soc.getInetAddress() + ":" +
soc.getPort();
        show(Message);
        //Creating both input and output Data streams and passing the input and output stream into the
created object streams
       try{
          in=new ObjectInputStream(soc.getInputStream());
          out=new ObjectOutputStream(soc.getOutputStream());
       }
```

```
   catch(IOException ex)
   {
      show("Exception arised while creating new I/O Streams: "+ex);
      return false;
   }
   //starts a thread to listen to server
   new LFromServer().start();
   //Sending username to Server in string format
   try
   {
       out.writeObject(uname);
   }
   catch(IOException ex)
   {
      show("Exception caused while logging in : "+ex);
      disconnect();
      return false;
   }
   return true;
 }
 //Main Method
 public static void main(String[] args)
 {
    //Default port used here is 5000 if user doesnt specify port to start server.
    int portnum=5000;
    //Default parameters
    String serverAdd="localhost";
```

```
String user= "anonymous";
Scanner sin=new Scanner(System.in);


System.out.println("Enter a name as a username:");
user=sin.nextLine();
switch(args.length)
{
   case 3:
      //Passes the username portno and server address as arguements
      serverAdd=args[2];
   case 2:
      try
      {
        //Passing username and portno as args
        portnum=Integer.parseInt(args[1]);
      }
      catch(Exception e)
      {
         System.out.println("Wrong Port number");
         System.out.println("Wrong Format of taking arguements");
         return;
      }
   case 1:
    //Passes only username as args
     user = args[0];
   case 0:
      break;
```

```
            default:
               System.out.println("Correct Format is java VChatClient user portnum serverAdd");
            return;
        }
        //Client object
        VChatClient vcl= new VChatClient(serverAdd,portnum,user);
        //using the client object to start client service and returns back when server instance isnt running
        if(!vcl.start())
            return;
        System.out.println("\nHOLAAA! Welcome To VCHATT!!! CHAT ROOM.");
        System.out.println("Select Service of your choice:");
        System.out.println("1) Send a broadcast message to all clients by just typing the message");
        System.out.println("2) Type '$username<space>message' to send a  private message to a desired 
    client");
        System.out.println("3) Type 'WHOISIN' to find out the active users present in chat room");
        System.out.println("4) Type 'U' to display the number of users connected to server");
        System.out.println("5) Type 'LOGOUT' to logoff from server");

        //Infinite loop which runs as long as the client wishes to stay and exits on LOGOUT
        while(true)
        {
            System.out.print("> ");
            //Users Choice or action
            String msg=sin.nextLine();
            //Compares the entered choice by user and sends passes to the Chat class method on match
            if(msg.equalsIgnoreCase("LOGOUT")) {
                                                vcl.sendMsg(new Chat(Chat.Logout, ""));
```

```
                    break;
                }

                else if(msg.equalsIgnoreCase("WHOISIN")) {
                    vcl.sendMsg(new Chat(Chat.WHOISIN, ""));

                }

                else if(msg.equalsIgnoreCase("U")){
                    vcl.sendMsg(new Chat(Chat.UserCount, ""));
                }
                else {
                    vcl.sendMsg(new Chat(Chat.MSG, msg));
                }
            }
            //Closing scanner class
            sin.close();
            //Disconnects the client
            vcl.disconnect();


}
//A class which extends Thread and waits for a response from server
class LFromServer extends Thread {

            public void run() {
                while(true) {
```

```
                                        try {


                                                  String msg = (String) in.readObject();

                                                  System.out.println(msg);

                                                  System.out.print("> ");

                                        }

                                        catch(IOException e) {

                                                  show(not + "User logged out successfully! closing service!: "
+ e + not);

                                                  break;

                                        }

                                        catch(ClassNotFoundException e2) {

                                        }

                              }

                    }

          }


}
```

VChatServer.java

```
import java.io.*;

import java.net.*;

import java.text.SimpleDateFormat;

import java.util.*;

public class VChatServer {

          //Unique id for every new connection

          private static int uId;

          //an arraylist which stores the users
```

```
private ArrayList<Client> al;

// timestamp

private SimpleDateFormat datefmt;

//port no

private int port;

//a flag to check if server is running or not

private boolean continueflw;

private String not = " ***!! ";


//Server constructor

public VChatServer(int port) {

        this.port = port;

        //Setting the date format

        datefmt = new SimpleDateFormat("HH:mm:ss");

        al = new ArrayList<Client>();

}

// a method to start server service

public void start() {

        //Setting the flag true to mark the continuation of service

        continueflw= true;

        try

        {

                //Creating a server socket for server

                ServerSocket ss = new ServerSocket(port);


                while(continueflw)

                {
```

```
                              show("Server waiting for Clients on port " + port + ".");
                              //Accepts request from client via accept() method
                              Socket soc= ss.accept();
                              //breaks the flow when server is terminated
                              if(!continueflw)
                                        break;
                          // Client thread created on successfull connection
                              Client client= new Client(soc);
                              // Add the connected user to arraylist
                              al.add(client);
//Start the client thread
                              client.start();
                    }
                    //If intentionally stopping the server
                    try {
                              //Closing Server scoket
                               ss.close();
                              for(int i = 0; i < al.size(); ++i) {
                                        Client tc = al.get(i);
                                        try {
                                                  //Closing all data streams
                                        tc.in.close();
                                        tc.out.close();
                                        tc.soc.close();
                                        }
                                        catch(IOException e) {
                                        }
                    }
                }
                catch(Exception e) {
                    show("Exception arised due to interruption!Closing the server and
clients: " + e);
                }
            }
            catch (IOException e) {
        String msg = datefmt.format(new Date()) + " Exception arised on new ServerSocket: " + e +
"\n";
                show(msg);
            }
    }

    //Method to stop server
    protected void stopserver() {
            continueflw = false;
            try {
                new Socket("localhost", port);
            }
            catch(Exception e) {
                System.out.println("See ya! Server is stopped!!"+ e);
            }
    }

    //a method to display a message along with timestamp
    private void show(String msg) {
        String time = datefmt.format(new Date()) + " " + msg;

        System.out.println(time);

}


//Method to broadcast message to all clients

private synchronized boolean broadcast(String message) {

        //Timestamp

        String time = datefmt.format(new Date());

        //Using this string array for private message operation

        String[] w = message.split(" ",3);

        //Flag to set up to true  private message option is selected

        boolean isPrivate = false;

        //When the first character matches the dollar sign the private flag is set to true

        if(w[1].charAt(0)=='$')

                isPrivate=true;

        if(isPrivate==true)

        {

                //Separates the symbol and username by using the substring function

                String tocheck=w[1].substring(1, w[1].length());


                message=w[0]+w[2];

                String messagefinal = time + " " + message + "\n";

                boolean found=false;

                //finding the username from the list and match with the string

                for(int y=al.size(); --y>=0;)

                {

                        Client ct1=al.get(y);
                        String check=ct1.getUsername();

                        if(check.equals(tocheck))

                        {

                                //Writing to client if server fails to remove it from list

                                if(!ct1.writeMsg(messagefinal)) {

                                        al.remove(y);

                                        show("Disconnected Client " + ct1.username + "
        removed from list.");

                                }

                                //Username match found and msg sent

                                found=true;

                                break;

                        }




                }

                //If match not found , return

                if(found!=true)

                {

                        return false;

                }

        }

        //Broadcast message case

        else

        {

                String messagefinal = time + " " + message + "\n";
```

```
				//broadcast message sent by client to all other clients
				System.out.print(messagefinal);

				for(int i = al.size(); --i >= 0;) {
					Client ct = al.get(i);
					if(!ct.writeMsg(messagefinal)) {
						al.remove(i);
						show("Disconnected Client is " + ct.username + " removed 
from list.");
					}
				}
			}
			return true;


		}

		synchronized void removeClient(int id) {

			String disconnectedClient = "";
			for(int i = 0; i < al.size(); ++i) {
				Client ct = al.get(i);
				if(ct.id == id) {
					disconnectedClient = ct.getUsername();
					al.remove(i);
					break;
				}
		}
		broadcast(not + disconnectedClient + " has left the chat room." + not);
	}

	public static void main(String[] args) {
		//Default server parameters
		int portNumber = 5000;
		switch(args.length) {
			case 1:
				try {
					portNumber = Integer.parseInt(args[0]);
				}
				catch(Exception e) {
					System.out.println("Invalid port number.");
					System.out.println("Wrong arguement format");
					return;
				}
			case 0:
				break;
			default:
				System.out.println("Correct format is: > java VChatServer 
portNumber");
				return;

		}
		//Server object
		VChatServer server = new VChatServer(portNumber);
```

```
                //Starting server service

                server.start();

        }

//Client class used to create client threads

        class Client extends Thread {

                //client socket

                Socket soc;

                //Object streams

                ObjectInputStream in;

                ObjectOutputStream out;

                //unique id

                int id;

                //username , chat object to get type of option selected,timestamp

                String username;

                Chat chat;

                String date;


                // Client Constructor

                Client(Socket soc) {


                        id = ++uId;

                        this.soc = soc;

                        System.out.println("Client Thread trying to create Object I/O Streams");

                        //Creating the data object streams

                        try

                        {

                                out = new ObjectOutputStream(soc.getOutputStream());
```

```
                    in = new ObjectInputStream(soc.getInputStream());
                    //reading username from socket input stream
                    username = (String) in.readObject();
                    //Broadcast message sent to all when a new user joins
                    broadcast(not + username + " has joined the chat room." + not);
                }
                catch (IOException e) {
                    show("Exception arised while creating new I/O Streams: " + e);
                    return;
                }
                catch (ClassNotFoundException e) {
                }
 date = new Date().toString() + "\n";
            }

            public String getUsername() {
                return username;
            }

            public void setUsername(String username) {
                this.username = username;
            }
//run() method to the run the client thread continuously
            public void run() {
                boolean continueflw = true;
                //Counter to count the users
                int count=0;
```

```
                    while(continueflw) {
                        try {
                            //Other than username rest all data entered are passed to
chat class for setting and getting them
                            //Hence reading the data from data input streams as chat
objects
                            chat = (Chat) in.readObject();
                        }
                        catch (IOException e) {
                            show(username + " Exception reading Streams: " + e);
                            break;
                        }
                        catch(ClassNotFoundException e2) {
                            break;
                        }
                        //Getting the message from the created chat object
                        String message = chat.getMSG();
            //Switch case method which performs the option selected by user
                        switch(chat.getType()) {
            //When it is a message type
                        case Chat.MSG:
                            boolean confirmation =  broadcast(username + ": " +
message);
                            if(confirmation==false){
                                String msg = not + "I am afraid there is no such user
in the room." + not;
                                writeMsg(msg);
                            }
```

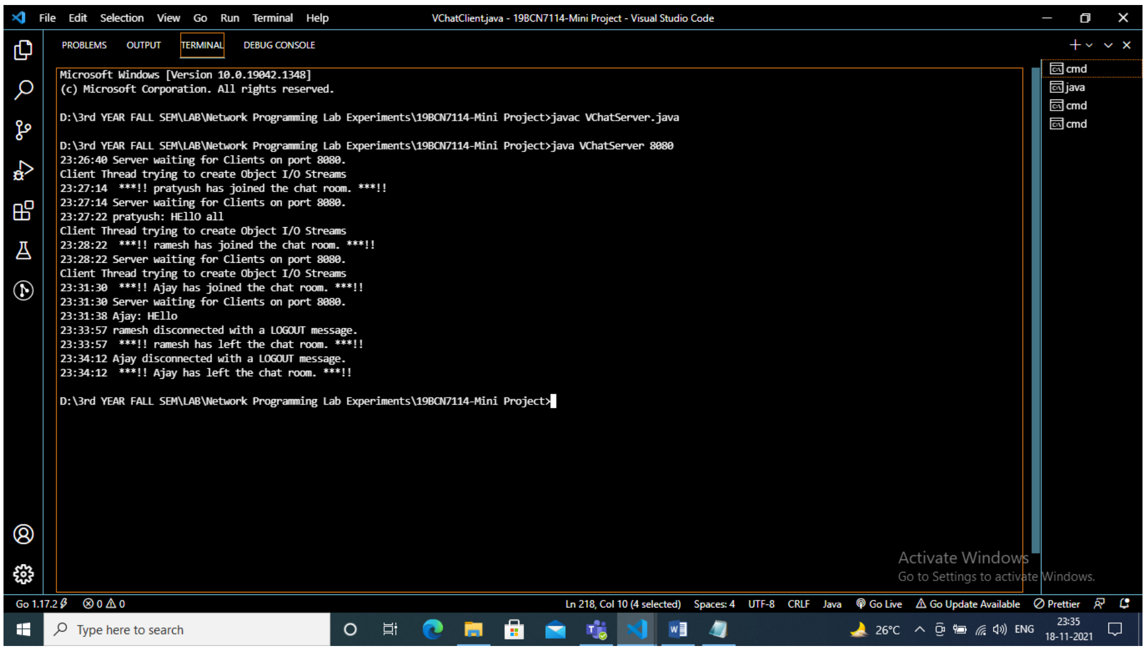
```
                                        break;
                                        //Logout case
                                case Chat.Logout:
                                        show(username + " disconnected with a LOGOUT
message.");
                                        continueflw = false;
                                        break;
                                        //names of active users in chat room
                                case Chat.WHOISIN:
                                        writeMsg("List of the users connected at " +
datefmt.format(new Date()) + "\n");


                                        for(int i = 0; i < al.size(); ++i) {
                                                Client ct = al.get(i);
                                                writeMsg((i+1) + ") " + ct.username + " since " +
ct.date);
                                        }
                                        break;
                                        //user count case
                                case Chat.UserCount:
                                  writeMsg("Number of users connected at" + datefmt.format(new
Date()) + "\n");
                                  for(int i = 1; i < al.size()+1; ++i) {
                                         count++;
                                  }
                                        writeMsg("Total numbers of users connected to server:
"+(count)+"\n");
```

```
                    count=0;
                  break;
                }
            }
            //Out of this loops means client is disconnected and client is removed from list
            removeClient(id);
            close();
        }

        // closes socket and streams
        private void close() {
            try {
                if(out != null)
                {
                    out.close();
                }
            }
            catch(Exception e) {}
            try {
                if(in != null)
                {
                    in.close();
                }
            }
            catch(Exception e) {};
            try {
```

```
                        if(soc != null)
                        {
                                soc.close();
                        }
                }
                catch (Exception e) {}
        }

        //method to Write to Client output stream
        private boolean writeMsg(String msg) {
                //Checks if socket is connected or not,if not then it closes socket and streams
                if(!soc.isConnected()) {
                        close();
                        return false;
    }
                try {
                        out.writeObject(msg);
                }
                catch(IOException e) {
                        show(not + "Err!!! Failed to send message to " + username + not);
                        show(e.toString());
                }
                return true;
        }
     }
}
```

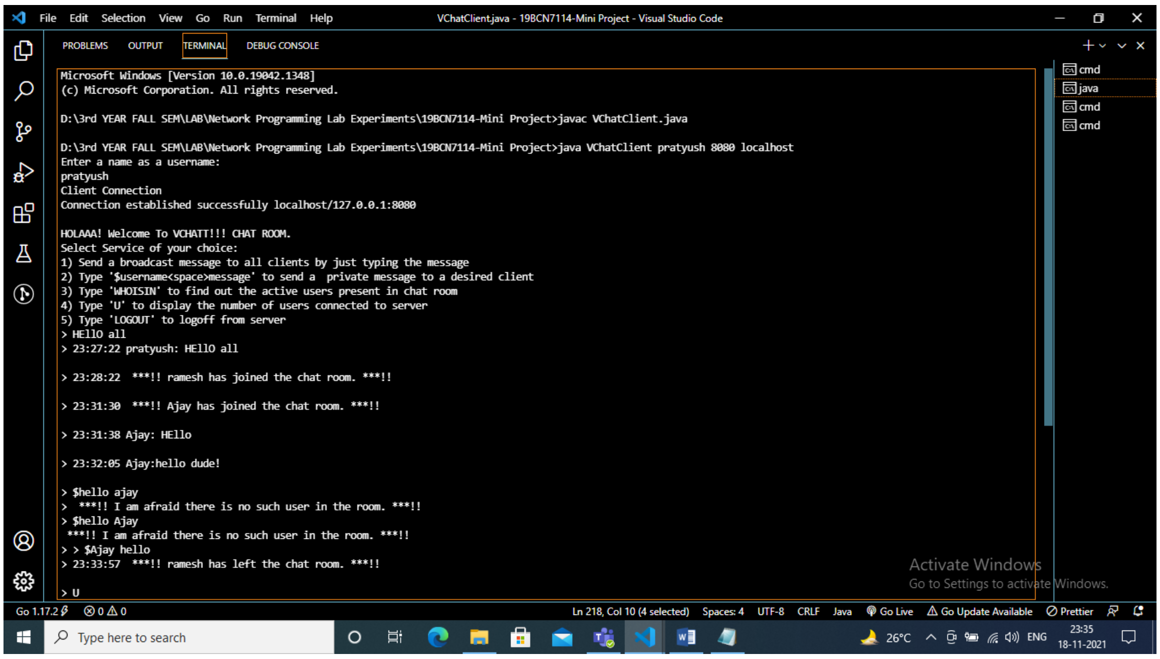
OUTPUTS

Server output

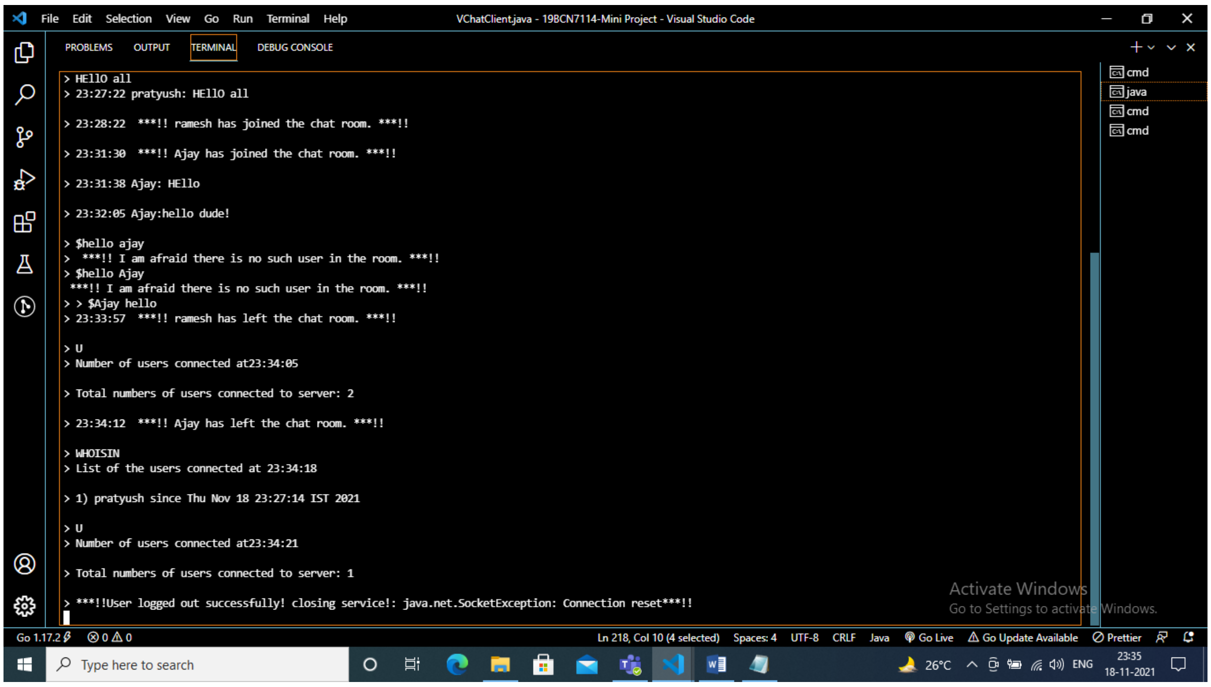

File Edit Selection View Go Run Terminal Help
VChatClient.java - 19BCN7114-Mini Project - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
cmd
java
cmd
cmd
Activate Windows
Go to Settings to activate Windows.
Go 1.17.2
Ln 218, Col 10 (4 selected) Spaces: 4 UTF-8 CRLF Java Go Live Go Update Available Prettier
Type here to search
26°C ENG 23:35 18-11-2021

```
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>javac VChatServer.java

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>java VChatServer 8080
23:26:40 Server waiting for Clients on port 8080.
Client Thread trying to create Object I/O Streams
23:27:14  ***!! pratyush has joined the chat room. ***!!
23:27:14 Server waiting for Clients on port 8080.
23:27:22 pratyush: HEllO all
Client Thread trying to create Object I/O Streams
23:28:22  ***!! ramesh has joined the chat room. ***!!
23:28:22 Server waiting for Clients on port 8080.
Client Thread trying to create Object I/O Streams
23:31:30  ***!! Ajay has joined the chat room. ***!!
23:31:30 Server waiting for Clients on port 8080.
23:31:38 Ajay: HEllo
23:33:57 ramesh disconnected with a LOGOUT message.
23:33:57  ***!! ramesh has left the chat room. ***!!
23:34:12 Ajay disconnected with a LOGOUT message.
23:34:12  ***!! Ajay has left the chat room. ***!!

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>
```

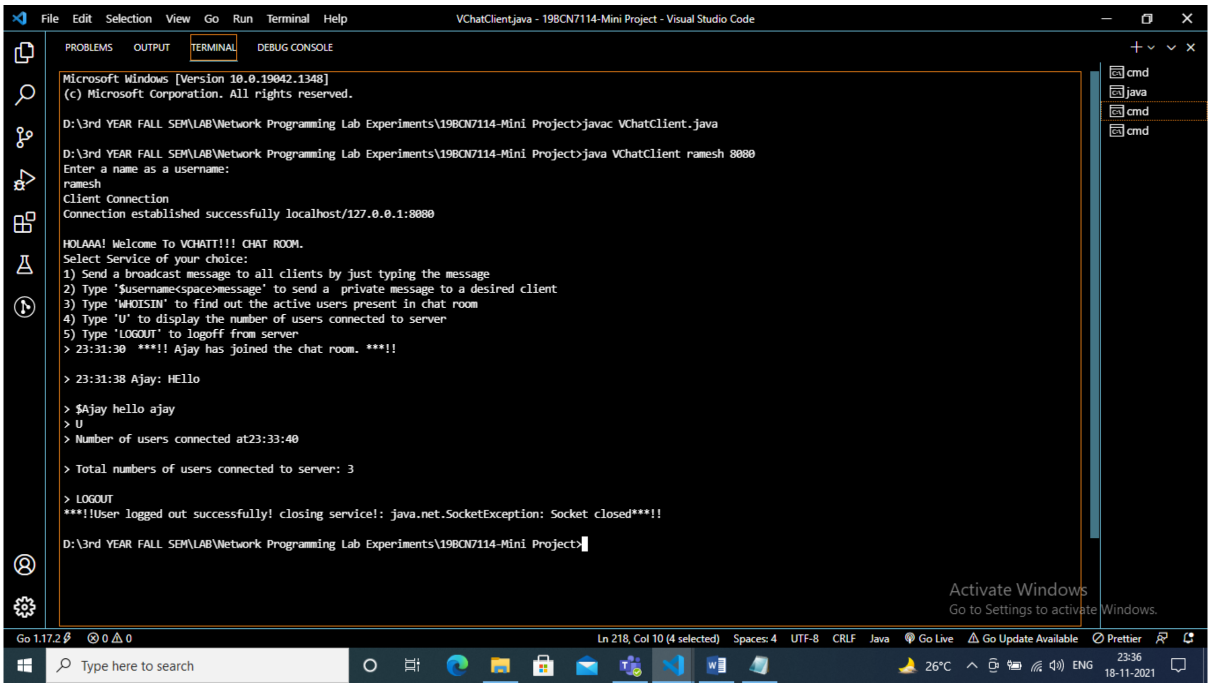
User 1 output


File Edit Selection View Go Run Terminal Help
VChatClient.java - 19BCN7114-Mini Project - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
cmd
java
cmd
cmd
Activate Windows
Go to Settings to activate Windows.
Go 1.17.2
Ln 218, Col 10 (4 selected) Spaces: 4 UTF-8 CRLF Java Go Live Go Update Available Prettier
Type here to search
26°C ENG 23:35 18-11-2021

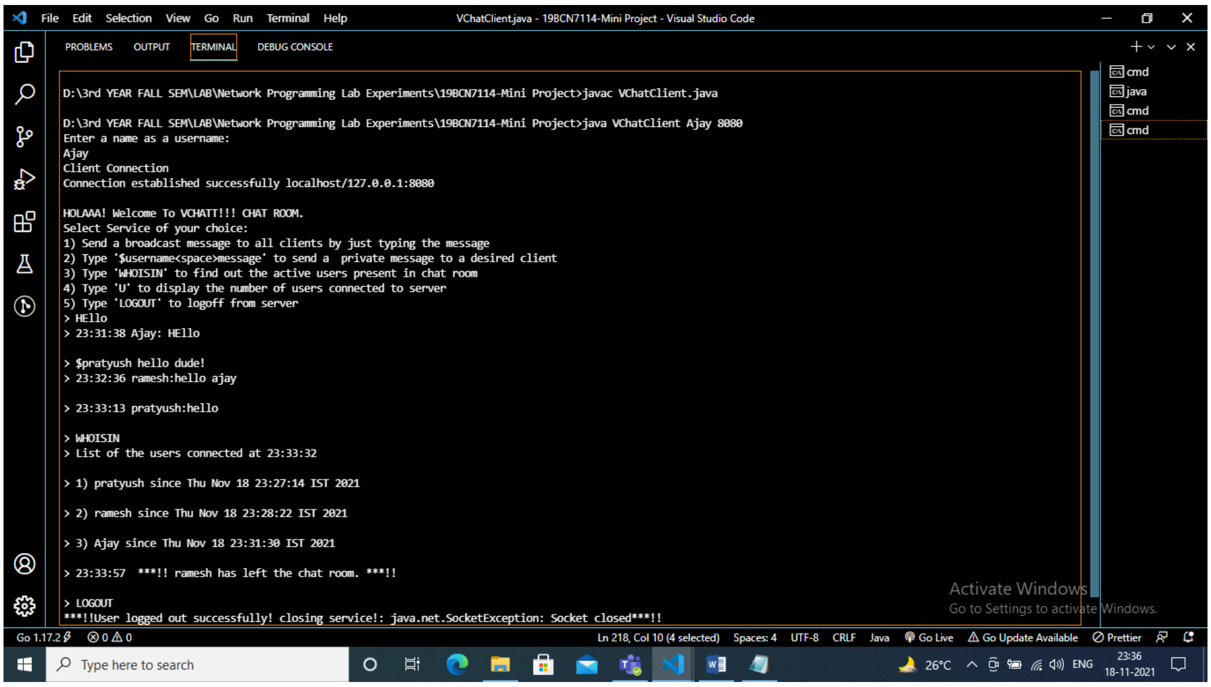
```
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>javac VChatClient.java

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>java VChatClient pratyush 8080 localhost
Enter a name as a username:
pratyush
Client Connection
Connection established successfully localhost/127.0.0.1:8080

HOLAAA! Welcome To VCHATT!!! CHAT ROOM.
Select Service of your choice:
1) Send a broadcast message to all clients by just typing the message
2) Type '$username<space>message' to send a  private message to a desired client
3) Type 'WHOISIN' to find out the active users present in chat room
4) Type 'U' to display the number of users connected to server
5) Type 'LOGOUT' to logoff from server
> HEllO all
> 23:27:22 pratyush: HEllO all

> 23:28:22  ***!! ramesh has joined the chat room. ***!!

> 23:31:30  ***!! Ajay has joined the chat room. ***!!

> 23:31:38 Ajay: HEllo

> 23:32:05 Ajay:hello dude!

> $hello ajay
>  ***!! I am afraid there is no such user in the room. ***!!
> $hello Ajay
 ***!! I am afraid there is no such user in the room. ***!!
> > $Ajay hello
> 23:33:57  ***!! ramesh has left the chat room. ***!!

> U
```


```
> HEllO all
> 23:27:22 pratyush: HEllO all

> 23:28:22  ***!! ramesh has joined the chat room. ***!!

> 23:31:30  ***!! Ajay has joined the chat room. ***!!

> 23:31:38 Ajay: HEllo

> 23:32:05 Ajay:hello dude!

> $hello ajay
>  ***!! I am afraid there is no such user in the room. ***!!
> $hello Ajay
 ***!! I am afraid there is no such user in the room. ***!!
> > $Ajay hello
> 23:33:57  ***!! ramesh has left the chat room. ***!!

> U
> Number of users connected at23:34:05

> Total numbers of users connected to server: 2

> 23:34:12  ***!! Ajay has left the chat room. ***!!

> WHOISIN
> List of the users connected at 23:34:18

> 1) pratyush since Thu Nov 18 23:27:14 IST 2021

> U
> Number of users connected at23:34:21

> Total numbers of users connected to server: 1

> ***!!User logged out successfully! closing service!: java.net.SocketException: Connection reset***!!
```

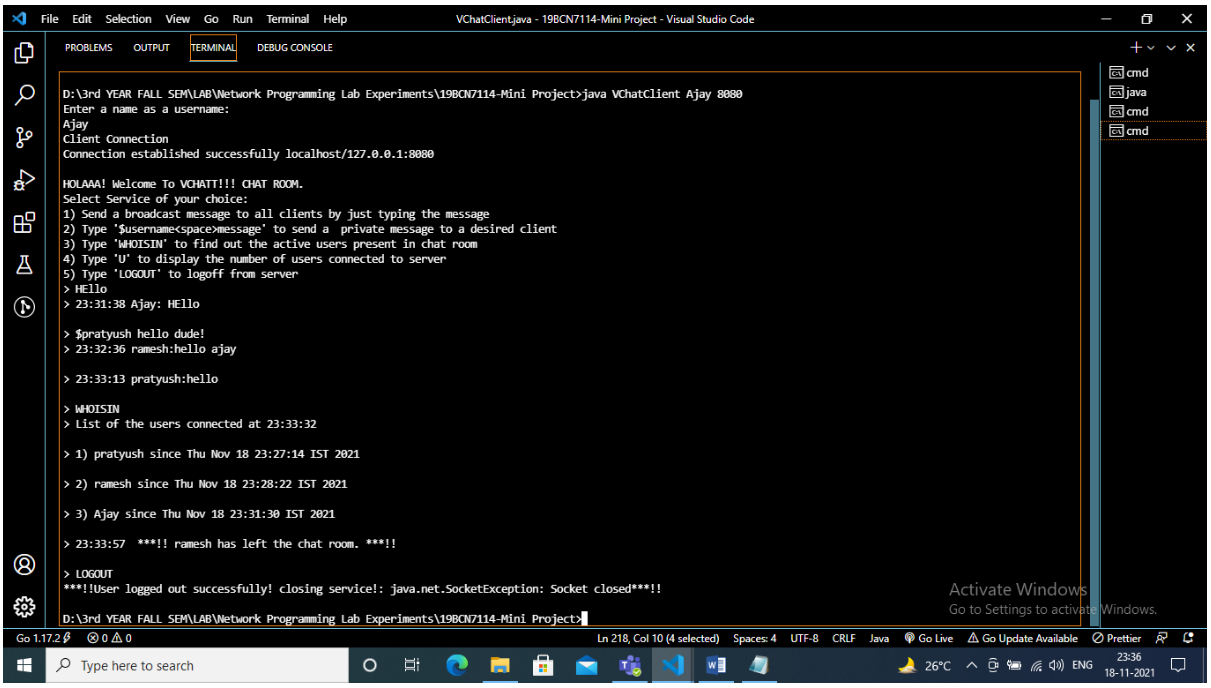
User 2 Output


```
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>javac VChatClient.java

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>java VChatClient ramesh 8080
Enter a name as a username:
ramesh
Client Connection
Connection established successfully localhost/127.0.0.1:8080

HOLAAA! Welcome To VCHATT!!! CHAT ROOM.
Select Service of your choice:
1) Send a broadcast message to all clients by just typing the message
2) Type '$username<space>message' to send a  private message to a desired client
3) Type 'WHOISIN' to find out the active users present in chat room
4) Type 'U' to display the number of users connected to server
5) Type 'LOGOUT' to logoff from server
> 23:31:30  ***!! Ajay has joined the chat room. ***!!

> 23:31:38 Ajay: HEllo

> $Ajay hello ajay
> U
> Number of users connected at23:33:40

> Total numbers of users connected to server: 3

> LOGOUT
***!!User logged out successfully! closing service!: java.net.SocketException: Socket closed***!!

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>
```

User 3 Output


File Edit Selection View Go Run Terminal Help
VChatClient.java - 19BCN7114-Mini Project - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
cmd
java
cmd
cmd
Activate Windows
Go to Settings to activate Windows.
Go 1.17.2 ⊗ 0 ⚠ 0
Ln 218, Col 10 (4 selected) Spaces: 4 UTF-8 CRLF Java Go Live Go Update Available Prettier
Type here to search
26°C ENG 23:36 18-11-2021

```
D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>javac VChatClient.java

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>java VChatClient Ajay 8080
Enter a name as a username:
Ajay
Client Connection
Connection established successfully localhost/127.0.0.1:8080

HOLAAA! Welcome To VCHATT!!! CHAT ROOM.
Select Service of your choice:
1) Send a broadcast message to all clients by just typing the message
2) Type '$username<space>message' to send a  private message to a desired client
3) Type 'WHOISIN' to find out the active users present in chat room
4) Type 'U' to display the number of users connected to server
5) Type 'LOGOUT' to logoff from server
> HEllo
> 23:31:38 Ajay: HEllo

> $pratyush hello dude!
> 23:32:36 ramesh:hello ajay

> 23:33:13 pratyush:hello

> WHOISIN
> List of the users connected at 23:33:32

> 1) pratyush since Thu Nov 18 23:27:14 IST 2021

> 2) ramesh since Thu Nov 18 23:28:22 IST 2021

> 3) Ajay since Thu Nov 18 23:31:30 IST 2021

> 23:33:57  ***!! ramesh has left the chat room. ***!!

> LOGOUT
***!!User logged out successfully! closing service!: java.net.SocketException: Socket closed***!!
```


File Edit Selection View Go Run Terminal Help
VChatClient.java - 19BCN7114-Mini Project - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
cmd
java
cmd
cmd
Activate Windows
Go to Settings to activate Windows.
Go 1.17.2 ⊗ 0 ⚠ 0
Ln 218, Col 10 (4 selected) Spaces: 4 UTF-8 CRLF Java Go Live Go Update Available Prettier
Type here to search
26°C ENG 23:36 18-11-2021

```
D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>java VChatClient Ajay 8080
Enter a name as a username:
Ajay
Client Connection
Connection established successfully localhost/127.0.0.1:8080

HOLAAA! Welcome To VCHATT!!! CHAT ROOM.
Select Service of your choice:
1) Send a broadcast message to all clients by just typing the message
2) Type '$username<space>message' to send a  private message to a desired client
3) Type 'WHOISIN' to find out the active users present in chat room
4) Type 'U' to display the number of users connected to server
5) Type 'LOGOUT' to logoff from server
> HEllo
> 23:31:38 Ajay: HEllo

> $pratyush hello dude!
> 23:32:36 ramesh:hello ajay

> 23:33:13 pratyush:hello

> WHOISIN
> List of the users connected at 23:33:32

> 1) pratyush since Thu Nov 18 23:27:14 IST 2021

> 2) ramesh since Thu Nov 18 23:28:22 IST 2021

> 3) Ajay since Thu Nov 18 23:31:30 IST 2021

> 23:33:57  ***!! ramesh has left the chat room. ***!!

> LOGOUT
***!!User logged out successfully! closing service!: java.net.SocketException: Socket closed***!!

D:\3rd YEAR FALL SEM\LAB\Network Programming Lab Experiments\19BCN7114-Mini Project>
```