

Anonymous Multi-Chat Application

A COURSE PROJECT REPORT

By

Chetas Shree Madhusudhan (RA2011003011120)
Ishaan Soman (RA2011003011154)
Rishabh Dharmik (RA2011003011129)
Eroth Sanju Rao(RA2011003011128)

Under the guidance of

Mrs. Rajalakshmi M

In partial fulfilment for the Course

of

18CSC302J - COMPUTER NETWORKS

in Computational Intelligence



FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chenpalattu District

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report Anonymous Multi-Chat Application is the bonafide work of Chetas Shree Madhusudhan (RA2011003011120) Ishaan Soman (RA2011003011154) ,Rishabh Dharmik (RA2011003011129) Eroth Sanju Rao(RA2011003011128) who carried out the project work under my supervision.

SIGNATURE

Mrs. Rajalakshmi M
Assistant Professor

Department of Computer Science and Engineering
SRM Institute of Science and Technology
Potheri, SRM Nagar, Kattankulathur,
Tamil Nadu 603203

SIGNATURE

Dr.E. Sasikala,
Course Coordinator

Associate Professor,
Data Science and Business Systems
SRM Institute of Science and Technology
Potheri, SRM Nagar, Kattankulathur,

TABLE OF CONTENTS

CHAPTERS	CONTENTS	PAGE NO.
1.	ABSTRACT	8
2.	INTRODUCTION	10
3.	REQUIREMENT ANALYSIS	12
4.	ARCHITECTURE & DESIGN	13
5.	IMPLEMENTATION	17
6.	EXPERIMENT RESULTS & ANALYSIS	24
	6.1. RESULTS	24
	6.2. RESULT ANALYSIS	25
	6.3. CONCLUSION & FUTURE WORK	26
7.	REFERENCES	27

Report Should contain minimum of 25 pages and maximum of 30 pages

ABSTRACT

The latest development of the Internet has brought the world into our hands. Everything happens through internet from passing information to purchasing something. Internet made the world as small circle. This project is also based on internet. This paper shows the importance of chat application in day today life and its impact in technological world. This project is to develop a chat system based on Java multithreading and network concept. The application allows people to transfer messages both in private and public way. It also enables the feature of sharing resources like files, images, videos, etc. This online system is developed to interact or chat with one another on the Internet. It is much more reliable and secure than other traditional systems available. Java, multi-threading and client-server concept were used to develop the web based chat application. This application is developed with proper architecture for future enhancement. It can be deployed in all private organizations like Colleges, IT parks, etc.

The chatting application has huge impact on day-to-day life. There is numerous chatting application available in this world. Each application has different additional features varying from other applications. These application organizations compete with each other and add some competing features during each release.

They have reached people much and have an impact on people's life. People find a better application from an available internet application which they feel much reliable and secure. Some of the available chatting applications that are available in these days are WhatsApp, Facebook, Instagram, Hike, etc... The above-mentioned applications have billion users all over the world. Those companies are one of the top companies in the world. They have higher revenue per year and have many employees for their organizations developing additional features to compete with other organizations during each release. These applications have different features and follows different ways to ensure security of their user data. Today a data theft is the major crime and most people are involved in it. There are many cases being filed these days about personal data loss. So, the organizations have to ensure the security from data loss by the third-party data crisis. The basic chatting system should involve both sending and receiving processes simultaneously. In this application both.

Introduction

The chatting application has huge impact on day-to-day life. There is numerous chatting application available in this world. Each application has different additional features varying from other applications. These application organizations compete with each other and add some competing features during each release. They have reached people much and have an impact on people's life. People find a better application from an available internet application which they feel much reliable and secure. Some of the available charting applications that are available in these days are WhatsApp, Facebook, Instagram, Hike, etc... The above-mentioned applications have billion users all over the world. Those companies are one of the top companies in the world. They have higher revenue per year and have many employees for their organizations developing additional features to compete with other organizations during each release. These applications have different features and follows different ways to ensure security of their user data. Today a data theft is the major crime and most people are involved in it. There are many cases being filed these days about personal data loss. So, the organizations have to ensure the security from data loss by the third-party data crisis. The basic chatting system should involve both sending and receiving processes simultaneously. In this application both

OBJECTIVE

- To implement a chat system for private network or organizations.
- To ensure security of the message and private data that will be shared over the network.
- To store confidential data in secure way.
- To develop a two-way communication system.
- To add additional features from other traditional systems that is available in the market.
- To allow both group chat and private chat.
- To enable easy and fast way of communication between people.
- To ensure unlimited data transfer without any restriction of size
- To make people get connected to others at any time, from anywhere.
- To have unlimited size to store message data.

3.3 REQUIREMENT SPECIFICATION

3.3.1 Hardware Requirements

Processor : 2.4 GHz Clock

Speed RAM : 1 GB

Hard Disk : 500 MB (Minimum free space)

3.3.2 Software Requirements

Operating System : Windows 7 or

above Platform : Java

Special Tools: Visual Studio CODE

ARCHITECTURE & DESIGN

This application has been implemented based on client/server model.

A. SERVER

A server may be a computer dedicated to running a server application. Organizations have dedicated computer for server application which has to be maintained periodically and has to be monitored continuously for traffic loads would never let them go down which affects the company's revenue. Most organizations have a separate monitoring system to keep an eye over their server so that they can find their server downtime before its clients. These server computers accept clients over network connections that are requested. The server responds back by sending responses being requested. There are many different server applications that vary based on their dedicated work. Some are involved for accepting requests and performing all dedicated works like business application servers while others are just to bypass the request like a proxy server. These server computers must have a faster Central processing unit, faster and more plentiful RAM, and bigger hard disc drive. More obvious distinctions include redundancy in power supplies, network connections, and RAID also as Modular design.

B. CLIENT

A client is a software application code or a system that requests another application that is running on dedicated machine called Server. These clients need not be

connected to the server through wired communication. Wireless communication takes place in this process. Client with a network connection can send a request to the server.

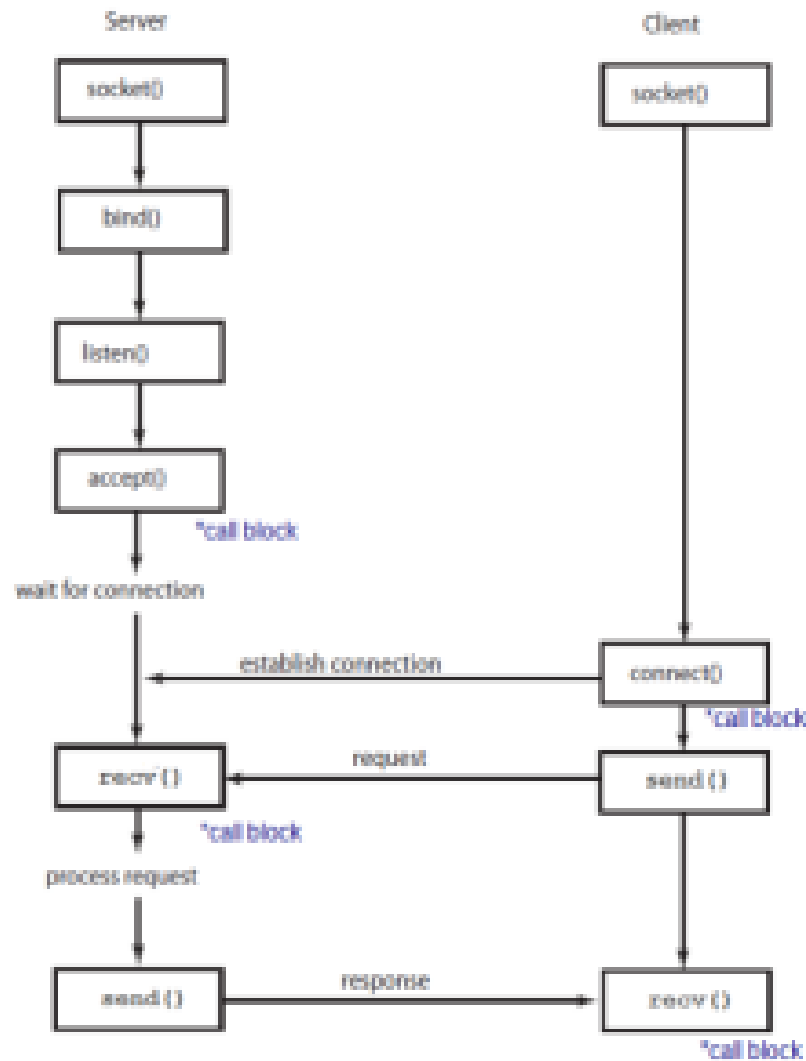


Fig. 1 Block Diagram

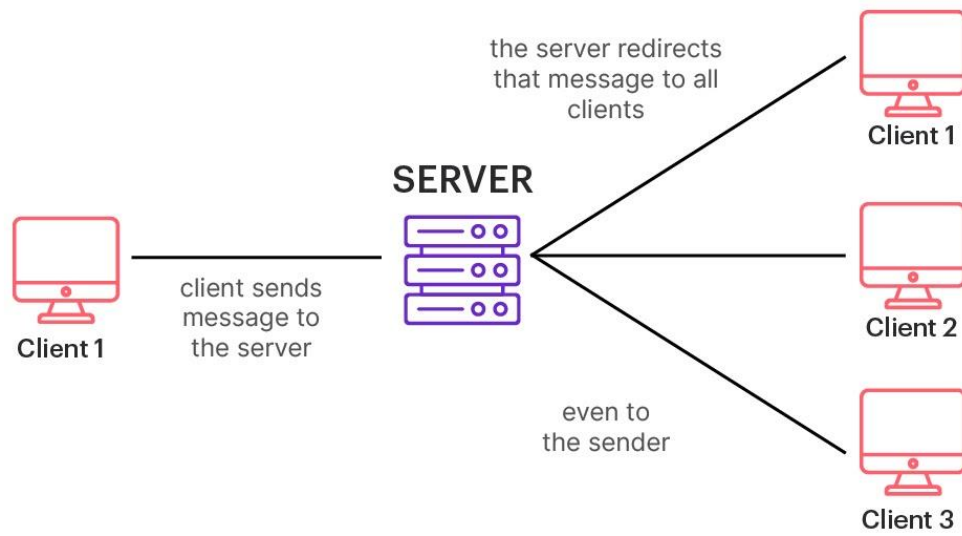


Fig. 2 Explanatory Diagram

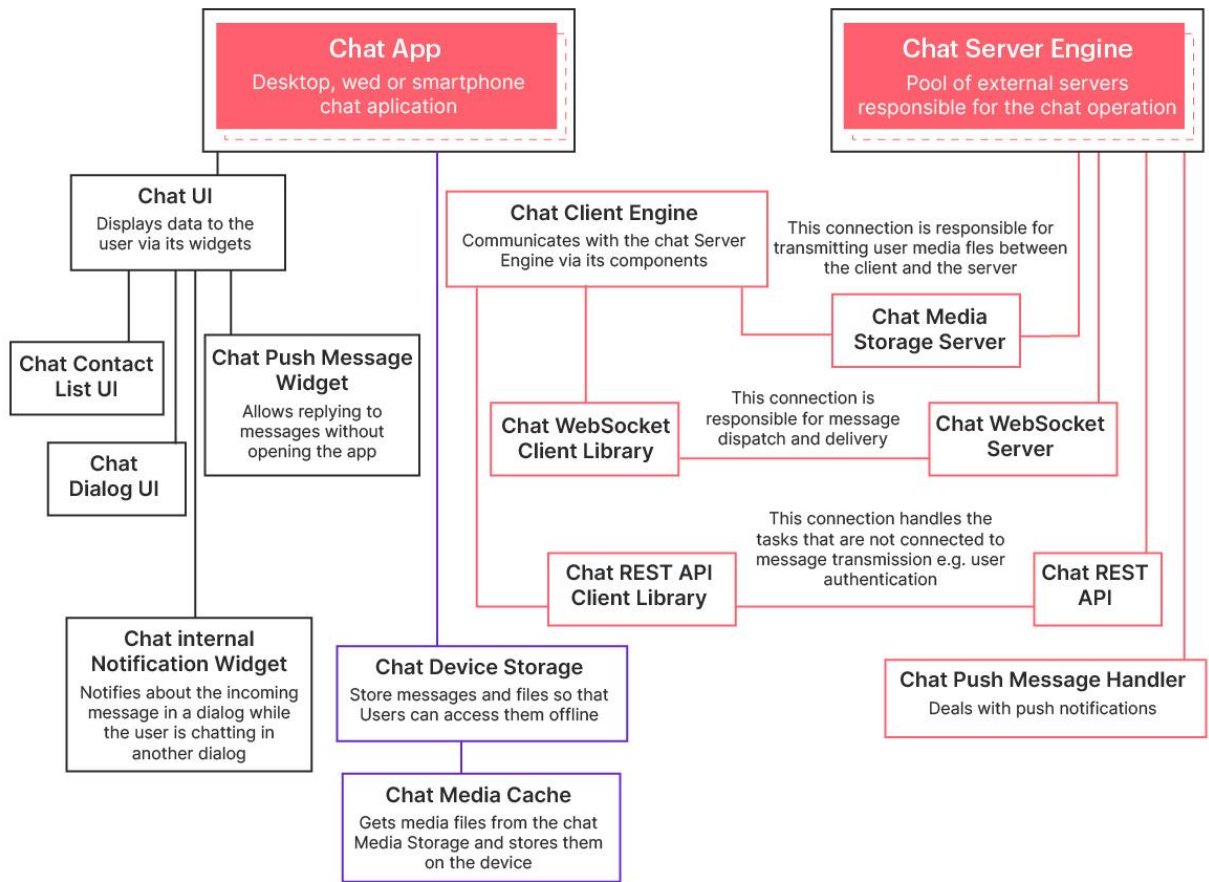


Fig. 3 Sequence Diagram

IMPLEMENTATION

1. A static Server socket is created in beginning which is then bind with host and port
2. After server instantiation Socket in particular host, it begins to listen in the particular port. Then the server is made to accept the request from the client through the particular port.
3. After starting the server, it can accept the requests from clients.
4. The socket is instantiated in client side to connect to the server.
5. A new Server Thread using socket is created to accept all the requests from multiple clients.
6. After accepting the request both read and write operation occurs simultaneously, clients who request the server can communicate with each other and share resources.
7. After finishing the communication the socket is closed both in the client and server side

CODE

Server.Java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class server {

    private static Socket client = null;
    private static ServerSocket socket = null;

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        //default port
        int port = 1222;

        if (args.length != 1) {
            System.err.println("Considering Port 1222");
            // System.exit(1);
        }
    }
}
```

```

else {

    port = Integer.parseInt(args[0]);

}

try {

    socket = new ServerSocket(port);

} catch(IOException e) {

    System.out.println(e.getMessage());

}

int countClients = 0;

while(true) {

    try {

        client = socket.accept();

        countClients++;

        ClientThread thr = new ClientThread(client,
countClients);

        thr.start();

    } catch (IOException e) {

        System.out.println(e.getMessage());

    }

}

}

class ClientThread extends Thread {

```

```

private Socket client = null;

private int clientNo = 0;


public ClientThread(Socket socket, int n) {

    clientNo = n;

    client = socket;

}


public void run() {

    try {

        System.out.printf("Client %d Connected\n", clientNo);


        BufferedReader in = new BufferedReader(new
InputStreamReader(client.getInputStream()));

        PrintWriter out = new PrintWriter(client.getOutputStream(),
true);


        String input;

        while ((input = in.readLine()) != null) {

            System.out.printf("Received from client %d: %s\n",
clientNo, input);

            out.println(input);

        }


        in.close();

        out.close();

        System.out.printf("Client %d Disconnected\n", clientNo);

        client.close();

```



```

    }

    catch (IOException e) {

        System.out.println(e.getMessage());

    }

}
}

```

client.Java

```

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.UnknownHostException;


public class client {


    public static void main(String[] args) {

        // TODO Auto-generated method stub

        //default host and port

        String host = "localhost";

        int port = 1222;


        if (args.length != 2) {

            System.err.println("Considering localhost and Port

1222");

            // System.exit(1);

        }

    }

}

```

```

else {

    host = args[0];

    port = Integer.parseInt(args[1]);

}

try {

    Socket socket = new Socket(host, port);

    BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

    PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);

    BufferedReader stdIn = new BufferedReader(new
InputStreamReader(System.in));

    String inp;

    while ((inp = stdIn.readLine()) != null) {

        out.println(inp);

        System.out.println("Received from Server: " +
reverseString(in.readLine()));

    }

    in.close();

    out.close();

    socket.close();

} catch (UnknownHostException e) {

    System.out.println(e.getMessage());

    System.exit(1);

} catch (IOException e) {

```

```

        System.out.println(e.getMessage());

        System.exit(1);
    }
}

public static String reverseString(String s) {
    StringBuilder rev = new StringBuilder(s.length() + 1);
    String[] words = s.split(" ");
    for (int i = words.length - 1; i >= 0; i--) {
        rev.append(words[i]);
        rev.append(" ");
    }
    rev.setLength(rev.length() - 1);
    String rever = rev.toString();
    return (rever);
}
}

```

Compile the code using the following lines:

```
$ javac server.java
```

```
$ javac client.java
```

Run the code:

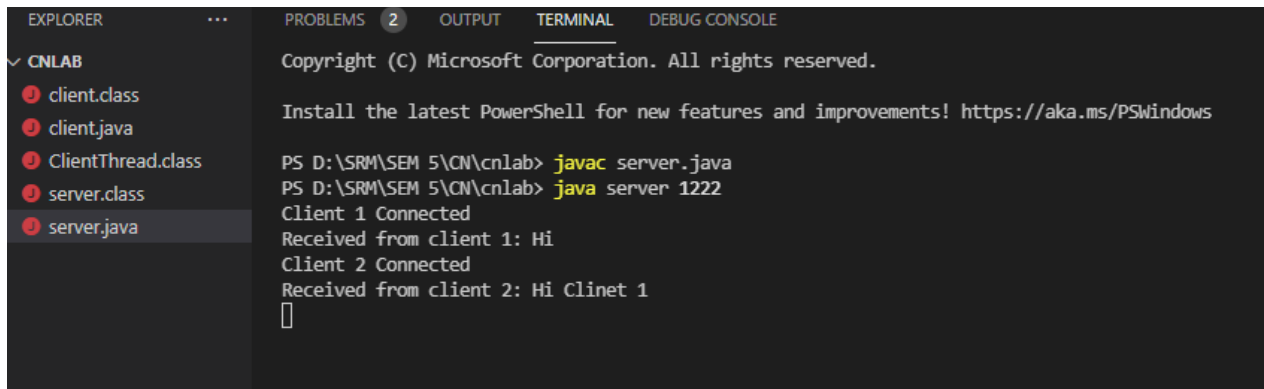
```
$ java server <PORT NUMBER>
```

```
$ java client localhost <PORT NUMBER>
```

6. EXPERIMENT RESULTS AND ANALYSIS

6.1 Results

Server



The screenshot shows the VS Code interface with the Explorer pane on the left displaying a project named 'CNLAB'. The file list includes 'client.class', 'client.java', 'ClientThread.class', 'server.class', and 'server.java'. The 'server.java' file is selected. The Terminal pane on the right shows the output of running 'server.java'. The output includes the standard Windows PowerShell copyright notice, a PowerShell update message, and the execution of 'javac server.java' and 'java server 1222'. The output shows 'Client 1 Connected', 'Received from client 1: Hi', 'Client 2 Connected', and 'Received from client 2: Hi Client 1'.

```
EXPLORER  ...  PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE

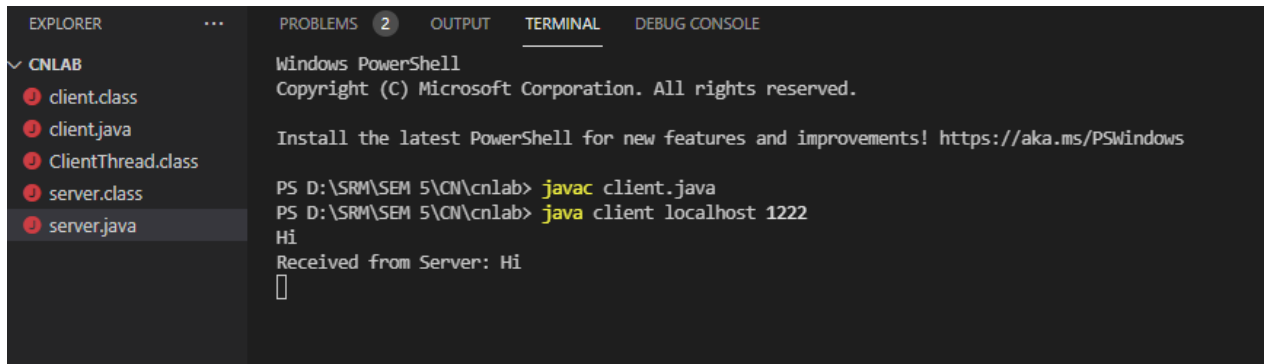
CNLAB
  client.class
  client.java
  ClientThread.class
  server.class
  server.java

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\SRM\SEM 5\CN\cnlab> javac server.java
PS D:\SRM\SEM 5\CN\cnlab> java server 1222
Client 1 Connected
Received from client 1: Hi
Client 2 Connected
Received from client 2: Hi Client 1
[]
```

Client 1



The screenshot shows the VS Code interface with the Explorer pane on the left displaying a project named 'CNLAB'. The file list includes 'client.class', 'client.java', 'ClientThread.class', 'server.class', and 'server.java'. The 'server.java' file is selected. The Terminal pane on the right shows the output of running 'client.java'. The output includes the standard Windows PowerShell copyright notice, a PowerShell update message, and the execution of 'javac client.java' and 'java client localhost 1222'. The output shows 'Hi' and 'Received from Server: Hi'.

```
EXPLORER  ...  PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE

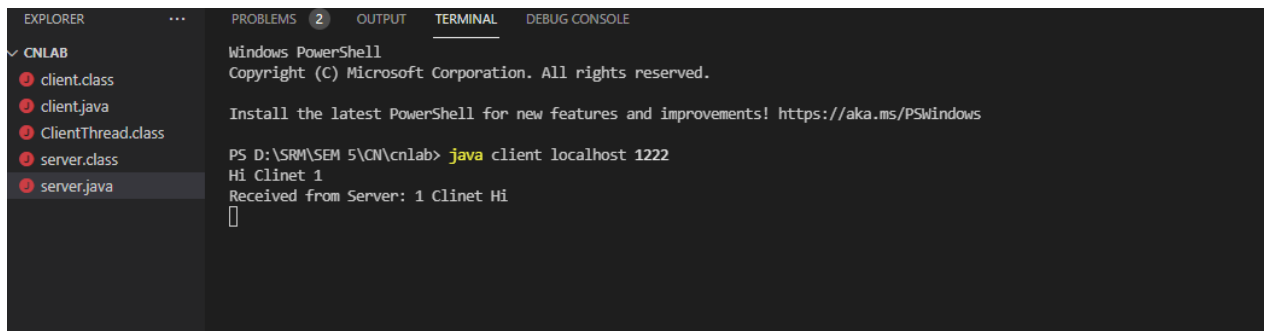
CNLAB
  client.class
  client.java
  ClientThread.class
  server.class
  server.java

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\SRM\SEM 5\CN\cnlab> javac client.java
PS D:\SRM\SEM 5\CN\cnlab> java client localhost 1222
Hi
Received from Server: Hi
[]
```

Client 2



```
EXPLORER  ...  PROBLEMS  2  OUTPUT  TERMINAL  DEBUG CONSOLE
CNLAB
  client.class
  client.java
  ClientThread.class
  server.class
  server.java

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\SRM\SEM 5\CN\cnlab> java client localhost 1222
Hi Clinet 1
Received from Server: 1 Clinet Hi
[]
```

6.2 RESULT ANALYSIS

The above figures show the output of the multi-user chat application. When any user sends the message it is sent to all the receivers along with the anonymous name.

Maximum user count is set to which the user can get connection. Once the client count exceeds the server intimates that server is busy and the client will wait until the server becomes idle. Any user can leave the chat box once it sends the message. Once the sender leaves it will be removed from the chat and all other users will be informed about it. Once the user leaves the chat box other users which are waiting for server to become idle will get a chance to communicate. Instead of sending message to all the users in the chat box message can be forwarded to particular user.

6.3 Conclusion and Future Works

Conclusion:

The chat application provides a better and flexible system for chatting. It is developed with recent advanced technologies in a way to provide a reliable system.

Main advantages of the system are instant messaging, real-world connectivity, adding security, group chat, etc. This application can find better need in the market for most of the organizations aim at having private applications for them.

Additional features will also be included in the system based on the public need which includes conference call, video chat. Location share, etc. based on the need

Future Works:

Further enhancements would be involved in the area of security, video call, large size transfer and some additional features that are required in the competing world. Other work is involving in implementation of the system in private networks

REFERENCES

- <https://www.ijeat.org/wp-content/uploads/papers/v9i5/E9578069520.pdf>
- <https://www.geeksforgeeks.org/multi-threaded-chat-application-set-1/>
- <https://gyawaliमित.medium.com/multi-client-chat-server-using-sockets-and-threads-in-java-2d0b64cad4a7>