

A COURSE PROJECT REPORT

By **Sharwan Kumar (RA2011031010034) Anand
Kunal Mishra (RA2011031010033) Saloni Smriti
(RA2011031010021) Riddhisatwa Ghosh
(RA2011031010004)**

Under the guidance of

Mrs. Thenmalar

In partial fulfilment of the Course

of

18CSC302J - COMPUTER NETWORKS

Networking and Communication Department with spec in IT



**FACULTY OF ENGINEERING AND
TECHNOLOGY SRM INSTITUTE OF SCIENCE
AND TECHNOLOGY**

Kattankulathur, Chengalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "Chat-bot Application" is the bonafide work of **Sharwan Kumar (RA2011031010034), Anand Kunal Mishra (RA2011031010033), Saloni Smriti (RA2011031010021), Riddhisatwa Ghosh (RA2011031010004)** who carried out the project work under my supervision.

SIGNATURE

Mrs. Thenmalar

Department of Networking and Communication

SRM Institute of Science and Technology

ABSTRACT

Chat refers to the process of communicating, interacting and/or exchanging messages over the Internet. It involves two or more individuals that communicate through a chat-enabled service or software. Chat may be delivered through text, audio or video communication via the Internet. A chat application has basic two components, via server and client. A server is a computer program or a device that provides functionality for other programs or devices. Clients who want to chat with each other connect to the server. The chat application we are going to make will be more like a chat room, rather than a peer-to-peer chat. So this means that multiple users can connect to the chat server and send their messages. Every message is broadcasted to every connected chat user.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honourable **Vice Chancellor Dr C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavours.

We would like to express my warmth of gratitude to our **Registrar Dr S. Ponnusamy**, for his encouragement We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr T. V. Gopal**, for bringing out novelty in all executions.

We would like to express our heartfelt thanks to the **Chairperson, School of Computing Dr Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express our sincere thanks to **Course Audit Professor Dr Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications and Course Coordinators** for their constant encouragement and support.

We are highly thankful for our Course project Faculty Mrs Thenmalar, Assistant Professor, NWC, for her assistance, timely suggestions and guidance throughout the duration of this course project. We extend my gratitude to our **HoD Dr Annapurani Panaiyappan**, NWC and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

| CHAPTERS | CONTENTS |
|----------|------------------------------------|
| | ABSTRACT |
| 1. | INTRODUCTION |
| 2. | REQUIREMENT ANALYSIS |
| 3. | ARCHITECTURE & DESIGN |
| 4. | IMPLEMENTATION |
| 5. | RESULTS AND DISCUSSIONS |
| 6. | CONCLUSION & FUTURE ENHANCEMENT |
| 7. | REFERENCES |
| 8 | INSTRUCTIONS |
| 9 | FEATURES |

INTRODUCTION

1. Scenario Description

A chat application is a type of software that enables users to communicate with each other. Applications can be anything from a chat room to an online forum. It allows users to communicate through text. It also lets them post messages and share information in real time. The twenty-first century is the age of technology and social media, and with the rise of social media, the number of applications has skyrocketed over the last decade. With the rise of online access and low-cost internet, life has become much easier to obtain things with a single click and accessing the functionality of the application. However, with the rising risk of data leakage and application hacking, security and privacy have become the most important factors for users in recent years. So, with this current motivation, we are creating a chat application using socket programming in JavaScript using node.js, and we have also enabled various features in the application with a strong emphasis on user privacy, security, and anonymity.

2. Problem Statement

- The goal of this mini project is to construct a chat application (Chat-Application) that includes a server and also allows users to talk with each other.
- To provide an instant messaging service that allows users to connect with each other in real-time.
- The project will be simple enough that even a beginner can use it.
- This project can be useful in organisational settings where employees can connect via LAN. The main goal of this project is to provide multi-chatting functionality via the network.

3. Innovative Ideas of Project

- GUI: This software has an easy-to-use GUI (Graphical User Interface), so any user with a basic understanding of computer operation can use it.
- Message Disappear: Messages will be disappeared after the user read them.
- Unlimited clients: "N" users can be connected without affecting the server's performance.

REQUIREMENTS

1. Functional Requirements

1. User Name: User have to enter their name before starting every new conversation.
2. Flat: User will select a flat they want to join, and can join it.
3. Send Message: User should be able to send instant message to any room and the users inthe flat can view the message and respond to it. There are some pre defined messages if user is not available it can alert the other user that the person whom you want to communicate is unavailable.

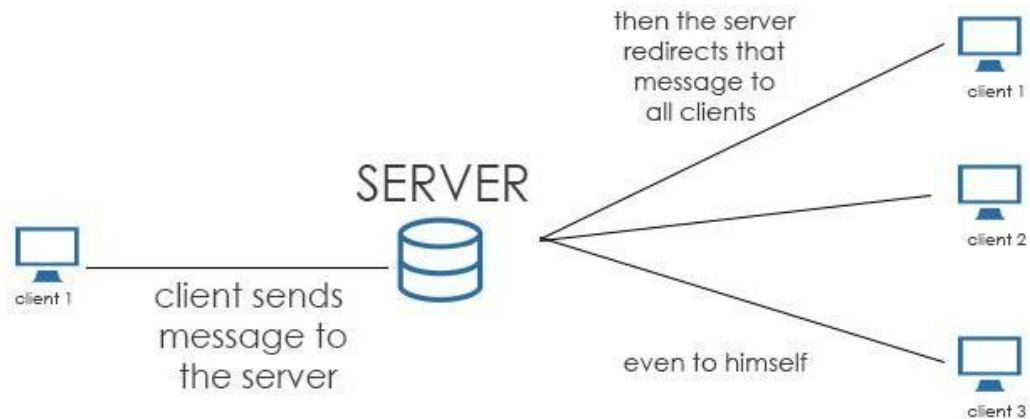
Non-Functional Requirements

1. Privacy: To protect privacy, exchanged messages between users should be encrypted and
whenever the messages has been read by the user ; automatically the messages will be deleted.
2. Robustness: Backup of every chat history is kept on distant database servers that aretotally secure.
3. Application performance: The application is quick and light on resources and very easy to use.

ARCHITECTURE AND DESIGN

Through G.U.I., this application communicates with the user. The user interface is straightforward, user-friendly, and self-explanatory.

Apps for messaging and chatting are becoming increasingly popular. People enjoy chatting, which is the obvious cause for this. It's the preferred way of communicating in a variety of situations, such as working with a co-worker or keeping tabs on a loved one.



Chat Client

Chat Client is a software that handles instant messaging and chat rooms. It is placed in a user's computer. It enables real-time communication. In addition, it can be integrated with programs that support instant messaging. It's an easy way to connect with other people. The user interacts with the chat client. The chat client, which can be used on a computer, a mobile device, or both, communicates with the operating system (i.e. your computer, browser or smartphone). The chat client sends a message to the chat server, which is the other important component, when you input a message and press send.

Chat Server

A chat server is a server instance set up to provide resources for the related chat service. The server accepts messages sent by the user of a chat client and forwards them to the recipient(s). The most popular chat servers are IRC and XMPP. The chat server is exactly what it sounds like—a server (or, more frequently, a number of servers) that houses all the programs, frameworks, and databases required for the chat app to function. This server, or group of servers, is in charge of receiving messages in a secure manner, determining who should get them, adding them to a queue, and finally sending them on to the recipient's chat client. A Web Socket server is one of the resources on the chat server.

CODE

Server:

```
package com.example.chatfull;

import android.util.Log;

import android. widget.Toast;

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 {

    ShowInfoActivity activity;

    private ServerSocket serverSocket;

    private String self_ip_address;

    private int self_port;

    User user;

    private boolean stop = false;

    public Server(ShowInfoActivity activity, String self_ip_address, int self_port) {

        this.activity = activity;

        this.self_port = self_port;

        this.self_ip_address = self_ip_address;
```

```

        Thread socketServerThread = new SocketServerThread();

        socketServerThread.start();
    }

    private class SocketServerThread extends Thread {

        @Override

        public void run() {

            try {

                // create ServerSocket using specified port

                serverSocket = new ServerSocket(self_port);

                while (stop == false) {

                    // block the call until connection is created and return Socket object

                    Log.e("SERVER", "WAITING");

                    Socket received_userSocket = serverSocket.accept();

                    Log.e("SERVER", "CONNECTED");

                    //Receive user credentials

                    BufferedReader input = new BufferedReader(new
InputStreamReader(received_userSocket.getInputStream()));

                    final String client_cred = input.readLine();

                    if (client_cred == null)

                        continue;

                    String client_ip = client_cred.substring(0, client_cred.indexOf(':'));

                    String client_port = client_cred.substring(client_cred.indexOf(':') + 1,
client_cred.indexOf('_'));

                    String client_name = client_cred.substring(client_cred.indexOf('_')+1);

                    user = new User(client_ip, Integer.parseInt(client_port));

                    user.setName(client_name);

```

```

        user.setId(client_cred);

        //MainActivity.userArrayList.add(user);

        //---Sending self name---

        try {

            String response_message = self_ip_address + ":" + self_port + "_" +
DialogViewActivity.me.getName();

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

            out.println(response_message);

        } catch (IOException e) {

            Log.e("INSIDE SERVER", "Could Not Send Self Credentials");

            e.printStackTrace();

        }

        //-----//

        activity.runOnUiThread(new Runnable() {

            @Override

            public void run() {

                Toast.makeText(activity.getApplicationContext(), "Connected To: " + client_cred,
Toast.LENGTH_LONG).show();

                //Move to next activity from main thread

                activity.setConnected(user);

            }

        });

        stop = true;

        break;

    }

    } catch (IOException e) {

        e.printStackTrace();

    }

```

```

        onDestroy();
    }
}

void onDestroy() {
    if (serverSocket != null) {
        try {
            Log.e("SERVER", "Closing Server");

            serverSocket.close();

            stop = true;

            Thread.interrupted();
        } catch (IOException e) {
            Log.e("SERVER", "Could Not Close Server");

            e.printStackTrace();
        }
    }
}
}

```

Client:

```

package com.example.chatfull;

import android.os.AsyncTask;

import android.util.Log;

import android.view.View;

import android.widget.Toast;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

```

```

import java.io.PrintWriter;

import java.net.Socket;

public class Client extends AsyncTask<Void, Void, String> {

    ConnectToUserActivity activity;

    private String dstAddress, serverResponse = "";

    private int dstPort;

    private Socket clientSocket = null;

    User user;

    Client(String dstAddress, int dstPort, ConnectToUserActivity activity) {

        this.dstAddress = dstAddress;

        this.dstPort = dstPort;

        this.activity = activity;

    }

    @Override

    protected String doInBackground(Void... arg0) {

        try {

            Log.e("CLIENT", "Before Connection");

            clientSocket = new Socket(dstAddress, dstPort);

            if (clientSocket != null) {

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

                out.println(ShowInfoActivity.getSelfIpAddress() + ":" + ShowInfoActivity.getSelfPort() + "_"
+ DialogViewActivity.me.getName());

                Log.e("CLIENT", "After Connection");

                user = new User(dstAddress, dstPort);

```

```

        //MainActivity.userArrayList.add(user);

    }

    try {

        BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

        serverResponse = input.readLine();

        String connected_name = serverResponse.substring(serverResponse.indexOf('_')+1);

        user.setName(connected_name);

        user.setId(serverResponse);

    } catch (IOException e) {

        e.printStackTrace();

        Log.e("CLIENT", "Could not read socket");

    }

    activity.setUser(user);

} catch (Exception e) {

    e.printStackTrace();

    serverResponse = e.getCause().toString();

} finally {

    if (clientSocket != null) {

        try {

            clientSocket.close();

        } catch (IOException e) {

            Log.e("CLIENT", "Could Not Close Client");

            e.printStackTrace();

        }

    }

}

return serverResponse;

}

```



```

@Override

protected void onPostExecute(String result) {

    activity.runOnUiThread(new Runnable() {

        @Override

        public void run() {

            activity.progressOverlay.setVisibility(View.INVISIBLE);

            Toast.makeText(activity.getApplicationContext(), serverResponse, Toast.LENGTH_LONG).show();

        }

    });

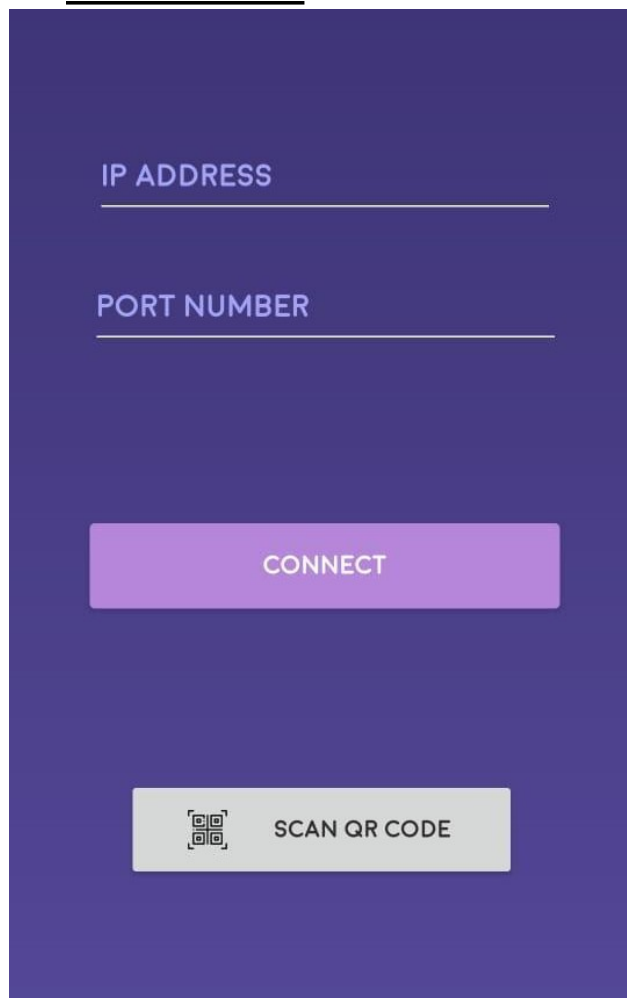
    super.onPostExecute(result);

}

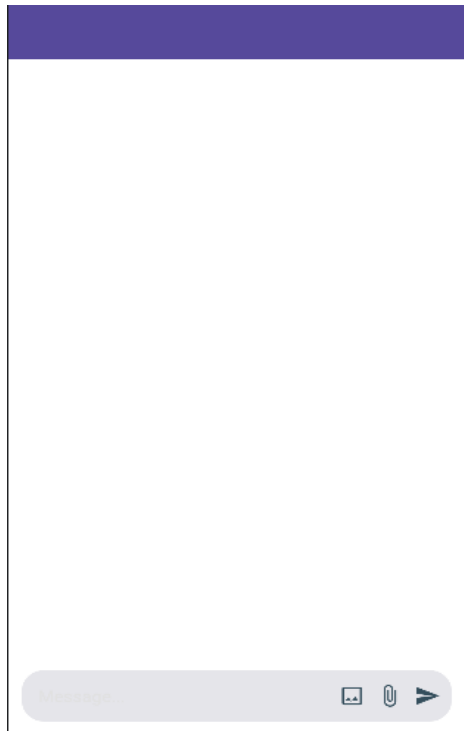
}

```

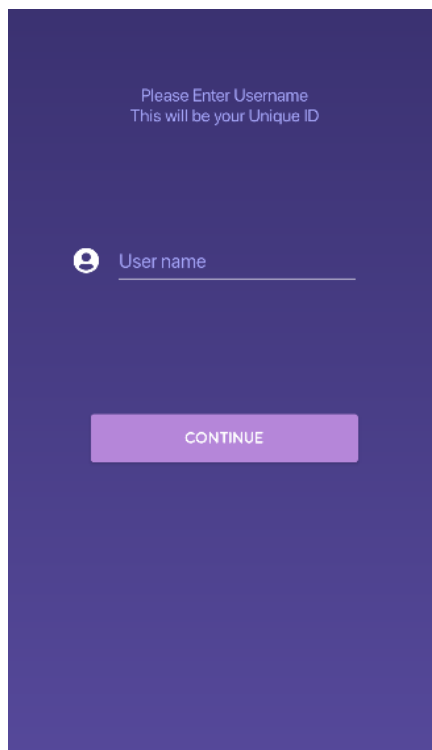
4.2 Screenshots



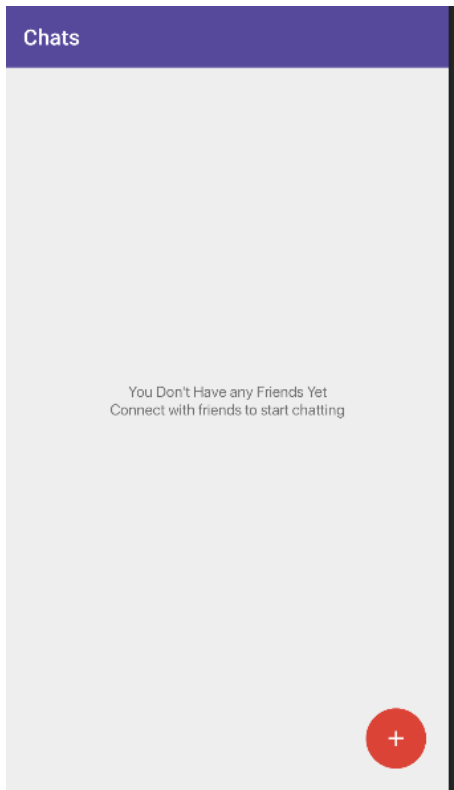
Start Chatting with Bot.



Enter your username.



The Chat App is for Mobiles and is fully responsive!



RESULTS AND DISCUSSION

The system's primary benefits are group chat, real-world connectivity, instant messaging, and additional security. A more efficient and adaptable approach to chatting is provided by the chatprogramme. Based on user demand, the system will also include extra features like conference calls and video chat. It is created using cutting-edge technologies in a way that offers a dependable system. Since most firms want to have private applications for them, this application can find greater demand in the market.

CONCLUSION AND FUTURE ENHANCEMENT

Although there are other chat apps that provide a similar function as this project, their user interfaces and usability were generally poor. Any and every software may always use some improvement. We just use text messages for communication right now. Both in interpersonal interactions and when using computers to engage with people, a good first impression is crucial. The goal of this project is to create a chat service Web application with a premium user interface. In the future, we might be enhanced with features like:

- o File Transfer
- o Video Message
- o Audio Call
- o Group Call

The project could also be included as a sub-part of a bigger project including chat-flat.

INSTRUCTIONS

1. First Connect both devices to the same wifi network.
2. Set username. **Username is CASE-SENSITIVE**. A username is used to store messages.
3. Click the "Show Information" button on one device.
4. Click the "Enter IP: Port" button on another device.
5. Scan The QR Code & click connect button.
6. Or enter the IP address and port number manually.
7. Start messaging.
8. To view any attachment the file must first be downloaded. Long press on file or image messages will save the file/image in the DOWNLOADS FOLDER and a notification will be shown in the notification bar of the device to open with external software.
9. **Long press** on text messages will copy the text to the clipboard.

FEATURES

- Text Messaging
- File sharing (any type of file including text, raw image, doc, pdf etc.)
- Image sharing
- Connect using QR Code Scanner
- Set background colour dynamically with custom colour picker.
- Show an image thumbnail in chat.
- Download and view shared files and images on your local device
- Dynamic online or offline status update of the opposite user.
- Chat is saved upon exit or going out of scope automatically. Manual saving is not necessary. The history is loaded from memory using the "Username+IP Address+Port" of the user. Change in either of these values will be detected as a different user.
- Chat is loaded dynamically to save memory consumption. Instead of loading entire message history at once

REFERENCES

- <https://tesseract-ocr.github.io/>
- <https://cryptography.io/en/latest/>
- <https://docs.python.org/3/library/tk.html>
- <https://pillow.readthedocs.io/en/stable/>
- <https://www.thepythoncode.com/article/send-receive-files-using-sockets-python>