

IMPLEMENTAION OF RTSP,FTP,TCP/IP PROTOCOL

*Report submitted to the SASTRA Deemed to be University
as the requirement for the course*

CSE302: COMPUTER NETWORKS

Submitted by

PRAVEENKUMAR B

(Reg.No:224003081, B. Tech Computer Science & Engineering)

December 2022

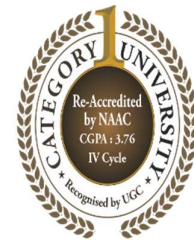


SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMBAKONAM, TAMIL NADU, INDIA – 612001

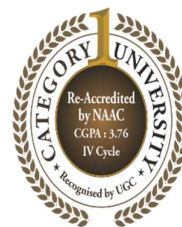


SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMBAKONAM – 612001

Bonafide Certificate

This is to certify that the report titled “**IMPLEMENTAION OF RTSP,FTP,TCP/IP PROTOCOL**” submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B. Tech is a bonafide record of the work done by **Shri. PRAVEENKUMAR B (Reg. No: 224003081, CSE)** during the academic year 2022-23, in the School of Computing.

Project Based Work Viva Voce held on _____

Examiner 1

Examiner 2

List of Figures

Figure No.	Title	Page No.
1.1	RTSP Protocol	2
1.2	FTP Protocol	3
1.3	TCP/IP Protocol	4
3.1	Server Initialization	44
3.2	Client-Side	44
3.3	Multi-Chat	45
3.4	Intimation of File Received	46
3.5	File Selection in the PC	46
3.6	File Received	47
3.7	Video Steaming	47

ABSTRACT

The primary goal of this project is to integrate RTSP, FTP, and TCP/IP in the Group Conversation Application. In this project, the video and audio in the communication are streamed using the RTSP protocol, and the chat is encrypted using end to end encryption (AES Algorithm). Any sort of file up to 2GB can be sent in a chat using TCP/IP (i.e similar to whatsapp). It is comparable to a group in which there is just one server and numerous clients. The chat's password is set up for login. These are features of this project.

KEYWORDS: RTSP,FTP,TCP/IP,Encryption,Decryption

Table of Contents

Title	Page No.
Bonafide Certificate	i
List of Figures	ii
Abstract	iv
1.1 Introduction	1
1.2 RTSP Protocol	1
1.3 Implementation of FTP Protocol	3
1.4 Implementation of TCP/IP Protocol	4
2 Source Code	6
3 Snapshots	44
4 Conclusion	48
5 References	49

CHAPTER 1

1.1 INTRODUCTION

Chat application is a feature or a program on the Internet to communicate directly among Internet users who are online or who were equally using the internet. Implementing a chat server application provides a good opportunity for a beginner to design and implement a network based system. The design is very simple. It is implemented in Java, since is easy to program in, it precludes the need to deal with low level memory management and includes powerful libraries for sockets and threads.

AES 256 based message Encryption.

Live (webcam) Video/Audio Transmision .

Supports File Transfer Upto 2 Gb.

All file format supported for File Transfer.

Uses TCP/IP Protocol for Message Transfer

1.2 RTSP PROTOCOL

Real Time Streaming Protocol (RTSP) is an application-level network communication system that transfers real-time data from multimedia to an endpoint device by communicating directly with the server streaming the data.

In the transport layer, RTP (Real-Time Protocol) is used to transmit the stream in real-time. The RTSP function is equivalent to the remote control of a streaming media server. IP cameras can use both TCP and UDP to transmit streaming content. However, it should be noted that UDP does not make any practical sense for this task.

RTSP uses the same concepts as basic HTTP, which makes it easily compatible with existing HTTP networks. The protocol also allows for a great deal of flexibility. Clients can request the features they want to use in an effort to find out if the media server supports them. Likewise, whoever owns the media can deliver media streams from multiple servers. The protocol is also designed to accommodate for future growth of the media so the media creator can amend the protocol moving forward if necessary.

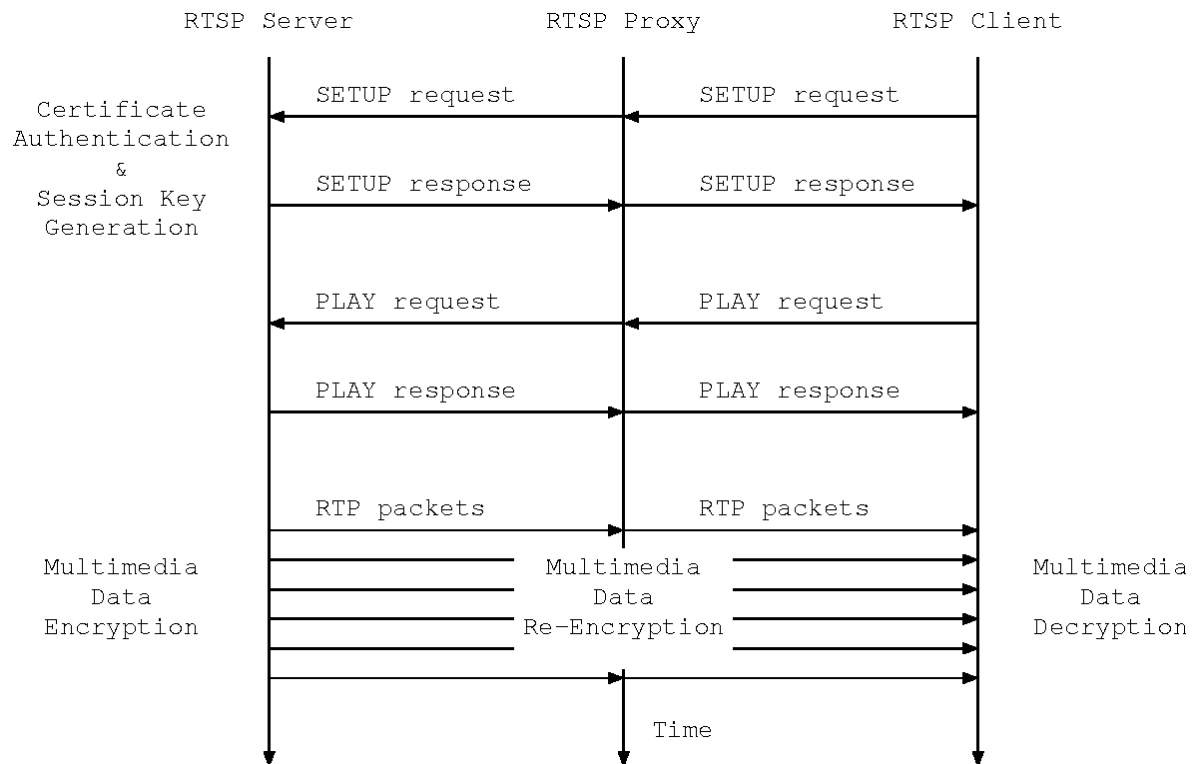


FIGURE 1.1 RTSP PROTOCOL

1.3 IMPLEMENTATION OF FTP PROTOCOL

The File Transfer Protocol is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client–server model architecture using separate control and data connections between the client and the server. Although transferring files from one system to another is very simple and straightforward, but sometimes it can cause problems. For example, two systems may have different file conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. FTP protocol overcomes these problems by establishing two connections between hosts. One connection is used for data transfer, and another connection is used for the control connection.

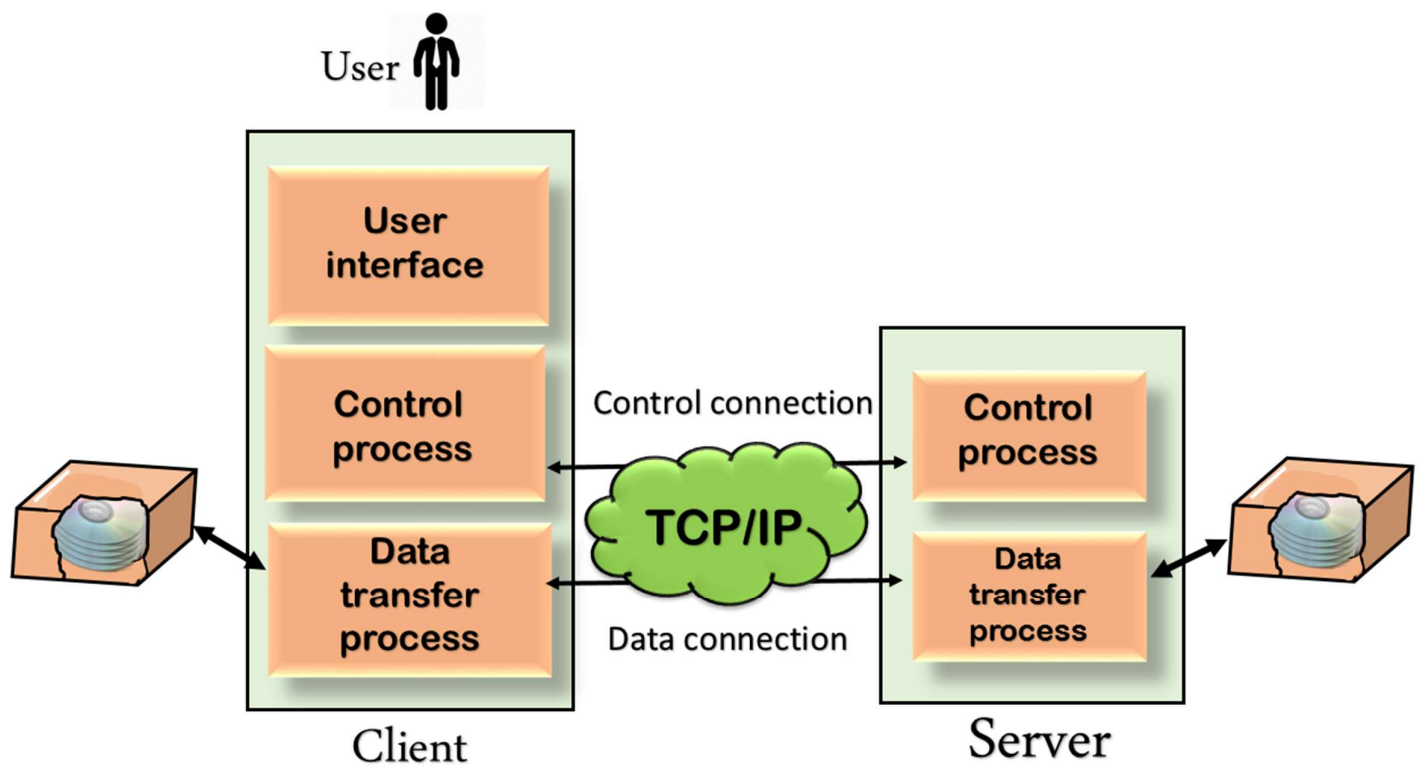


Figure 1.2:FTP PROTOCOL

1.4 IMPLEMENTATION OF TCP PROTOCOL

TCP is suited for applications that require high reliability, and transmission time is relatively less critical. It is used by other protocols like HTTP, HTTPs, FTP, SMTP, Telnet. TCP rearranges data packets in the order specified. There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent. TCP does Flow Control and requires three packets to set up a socket connection before any user data can be sent. TCP handles reliability and congestion control. It also does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.

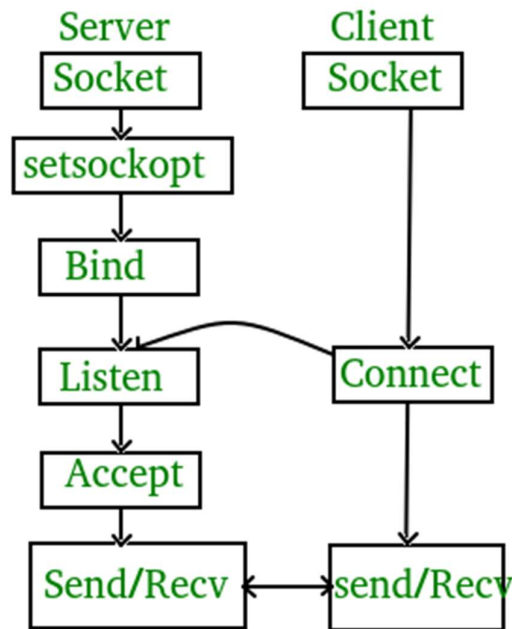


FIGURE 1.3:TCP/IP PROTOCOL

CHAPTER 2

SOURCE CODE

2.1 SOURCE CODE FOR SERVER

```
import java.net.Socket;
import java.net.SocketException;
import java.net.ServerSocket;
import java.util.ArrayList;
import java.util.Scanner;

import javax.swing.ImageIcon;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class Server {
    final static int PORT = 2000;

    static ArrayList<Socket> chatClientList;
    static ArrayList<Socket> videoClientList;
    static ArrayList<ObjectOutputStream> audioClientList;
    static String ENCRYPTED_SECRET_STRING;

    public static void main(String[] args) {
        chatClientList = new ArrayList<Socket>();
        videoClientList = new ArrayList<Socket>();
        audioClientList = new ArrayList<ObjectOutputStream>();
        int connectedClients = 0;
        ServerSocket chatServerSocket, videoServerSocket, audioServerSocket;
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter The Group Name :");
        String groupName = scan.nextLine();
        scan.close();
        try {

            chatServerSocket = new ServerSocket(PORT);
            videoServerSocket = new ServerSocket(PORT + 1);
            audioServerSocket = new ServerSocket(PORT + 2);
            new VideoServer(videoServerSocket).start();
            new AudioServer(audioServerSocket).start();
            System.out.println("Server Created with Port No: 2000 and Listening ...");

            while (true) {
                Socket client = chatServerSocket.accept();
                DataOutputStream dout = new DataOutputStream(client.getOutputStream());
```

```

        dout.writeUTF(groupName);
        connectedClients++;
        if (connectedClients == 1) {
            dout.writeUTF("RequestSecretText");
            ENCRYPTED_SECRET_STRING = new
DataInputStream(client.getInputStream()).readUTF();
        } else {
            dout.writeUTF(ENCRYPTED_SECRET_STRING);
        }
        System.out.println("Accepted new Client into the Server ");
        // System.out.println("Total Number of Connected Client :" + connectedClients);
        Server.chatClientList.add(client);
        new ClientListenThread(client).start();
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

```

class ClientListenThread extends Thread {
    Socket s;

    ClientListenThread(Socket s) {
        this.s = s;
    }

    public void run() {
        try {
            DataInputStream din = new DataInputStream(s.getInputStream());
            while (true) {
                String str = din.readUTF();
                if (str.startsWith("END")) {
                    s.close();
                    break;
                } else if (str.startsWith("FILE_TRANS")) {
                    byte bytes[] = new byte[Integer.parseInt(str.split("::")[2])];
                    din.readFully(bytes, 0, bytes.length);
                    for (Socket ss : Server.chatClientList) {
                        if (ss == s)
                            continue;
                        DataOutputStream dout = new DataOutputStream(ss.getOutputStream());
                        dout.writeUTF(str);
                        dout.write(bytes, 0, bytes.length);
                        dout.flush();
                    }
                    continue;
                }
            }
        }
        for (Socket s : Server.chatClientList) {

```

```

        DataOutputStream dout = new DataOutputStream(s.getOutputStream());
        dout.writeUTF(str);
    }
}
} catch (SocketException e) {
    System.out.println("Person Disconnected");
} catch (Exception e) {
    e.printStackTrace();
}
int i = Server.chatClientList.indexOf(s);
Server.chatClientList.remove(i);
}
}

```

```

class VideoServer extends Thread {

    ServerSocket videoServerSocket;

    VideoServer(ServerSocket ss) {
        videoServerSocket = ss;
    }

    public void run() {

        while (true) {
            try {
                Socket socket = videoServerSocket.accept();
                Server.videoClientList.add(socket);
                new VideoStreamThread(socket).start();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

class VideoStreamThread extends Thread {
    Socket s;

    VideoStreamThread(Socket socket) {
        s = socket;
    }

    public void run() {
        try {
            ImageIcon ic;
            ObjectInputStream oin = new ObjectInputStream(s.getInputStream());
            while (true) {

```

```

        ic = (ImageIcon) oin.readObject();
        if (ic != null && ic.getDescription() != null && ic.getDescription().equals("END")) {
            System.out.println("end received");
            s.close();
            break;
        } else {

            for (Socket c : Server.videoClientList) {
                // if(c==s) continue;
                ObjectOutputStream oout = new ObjectOutputStream(c.getOutputStream());
                oout.writeObject(ic);
                oout.flush();
            }
            if (ic != null && ic.getDescription() != null && ic.getDescription().equals("END_VIDEO"))
        }
        {
            oin = new ObjectInputStream(s.getInputStream());
        }
    }
}

} catch (Exception e) {
    e.printStackTrace();
}
int i = Server.videoClientList.indexOf(s);
Server.videoClientList.remove(i);
}
}

class AudioServer extends Thread {
    ServerSocket audioServerSocket;

    AudioServer(ServerSocket ss) {
        audioServerSocket = ss;
    }

    public void run() {
        try {
            while (true) {
                Socket s = audioServerSocket.accept();
                ObjectOutputStream out = new ObjectOutputStream(s.getOutputStream());
                Server.audioClientList.add(out);
                new AudioStreamThread(s, out).start();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

}

class AudioStreamThread extends Thread {
    Socket socket;
    private ObjectInputStream ois;
    private ObjectOutputStream out;

    AudioStreamThread(Socket s, ObjectOutputStream ot) {
        socket = s;
        out = ot;
    }

    public void run() {
        try {
            ois = new ObjectInputStream(socket.getInputStream());
            byte[] data = new byte[1024];
            while (true) {
                int dsize = ois.read(data);
                if (dsize == 1024) {
                    for (ObjectOutputStream oout : Server.audioClientList) {
                        oout.write(data, 0, dsize);
                        oout.reset();
                    }
                } else if (dsize == 512) {
                    System.out.println("[ SERVER ] : dsize-" + dsize + " Client Stopped.");
                    ois = new ObjectInputStream(socket.getInputStream());
                }
            }
        } catch (SocketException e) {
            System.out.println("Person Disconnected");
        } catch (Exception e) {
            System.out.println(e);
        }
        int i = Server.audioClientList.indexOf(out);
        Server.audioClientList.remove(i);
    }
}

```

2.2 SOURCE CODE FOR CLIENT:

```

import javax.crypto.BadPaddingException;

import javax.crypto.IllegalBlockSizeException;

import javax.sound.sampled.AudioFormat;

```



```

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.DataLine;

import javax.sound.sampled.SourceDataLine;

import javax.sound.sampled.TargetDataLine;

import javax.swing.*;

import javax.swing.border.EmptyBorder;


import java.net.*;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.awt.image.BufferedImage;

//import javax.swing.event.*;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.awt.event.*;

import java.awt.*;

import com.github.sarxos.webcam.*; // for getting webcam Videos


class Client extends JFrame {

```

```

static String IP_ADDRESS_STRING = "localhost";

static int PORT = 2000;

static String CURRENT_USER = "Client";

static String PASSWORD = "1234"; // FOR TESTING PURPOSES

static boolean isSetupDone;

static boolean runCam;

static Socket videoSocket;

static Socket audioSocket;

static JFrame videoFrame = new JFrame();

static final int VIDEO_HEIGHT = 640, VIDEO_WIDTH = 480;

static Encryption enc = new Encryption();

static Decryption dec = new Decryption();

/**
 *
 */

private static final long serialVersionUID = 1L;

Socket clientSocket;

JLabel groupName;

JButton send, fileSend, videoStream;

JTextField msg;

JPanel chat;

JScrollPane scrollPane;

JFileChooser jfc;

```

```

static {

    loginInterface();

}

private static void loginInterface() {

    Client.isSetupDone = false;

    JLabel nameLabel, ipLabel, portLabel, passwordLabel;

    JTextField nameTextField, ipTextField, portTextField;

    JPasswordField passwordTextField;

    JButton connect;

    JFrame frame = new JFrame();

    frame.setTitle("Set-UP");

    nameLabel = new JLabel("      Name :");

    ipLabel = new JLabel("IP Address :");

    passwordLabel = new JLabel(" Password :");

    portLabel = new JLabel("      Port :");

    nameTextField = new JTextField(15);

    ipTextField = new JTextField(15);

    portTextField = new JTextField(15);

    passwordTextField = new JPasswordField(15);

    connect = new JButton("Connect !");

    ipTextField.setText("localhost");

```

```

portTextField.setText(PORT + "");

Container contentPane = frame.getContentPane();

SpringLayout layout = new SpringLayout();

contentPane.setLayout(layout);

contentPane.add(nameLabel);

contentPane.add(nameTextField);

contentPane.add(ipLabel);

contentPane.add(ipTextField);

contentPane.add(portLabel);

contentPane.add(portTextField);

contentPane.add(passwordLabel);

contentPane.add(passwordTextField);

contentPane.add(connect);


// Name

layout.putConstraint(SpringLayout.WEST, nameLabel, 5, SpringLayout.WEST, contentPane);

layout.putConstraint(SpringLayout.NORTH, nameLabel, 5, SpringLayout.NORTH, contentPane);

layout.putConstraint(SpringLayout.WEST, nameTextField, 5, SpringLayout.EAST, nameLabel);

layout.putConstraint(SpringLayout.NORTH, nameTextField, 5, SpringLayout.NORTH,
contentPane);


// IP Address

layout.putConstraint(SpringLayout.WEST, ipLabel, 5, SpringLayout.WEST, contentPane);

layout.putConstraint(SpringLayout.NORTH, ipLabel, 5, SpringLayout.SOUTH, nameTextField);

layout.putConstraint(SpringLayout.WEST, ipTextField, 5, SpringLayout.EAST, ipLabel);

```

```

        layout.putConstraint(SpringLayout.NORTH,    ipTextField,    5,    SpringLayout.SOUTH,
nameTextField);

// Port

        layout.putConstraint(SpringLayout.WEST, portLabel, 5, SpringLayout.WEST, contentPane);

        layout.putConstraint(SpringLayout.NORTH, portLabel, 5, SpringLayout.SOUTH, ipTextField);

        layout.putConstraint(SpringLayout.WEST, portTextField, 5, SpringLayout.EAST, portLabel);

        layout.putConstraint(SpringLayout.NORTH,    portTextField,    5,    SpringLayout.SOUTH,
ipTextField);

// Password

        layout.putConstraint(SpringLayout.WEST,    passwordLabel,    5,    SpringLayout.WEST,
contentPane);

        layout.putConstraint(SpringLayout.NORTH,    passwordLabel,    5,    SpringLayout.SOUTH,
portTextField);

        layout.putConstraint(SpringLayout.WEST,    passwordTextField,    5,    SpringLayout.EAST,
passwordLabel);

        layout.putConstraint(SpringLayout.NORTH,    passwordTextField,    5,    SpringLayout.SOUTH,
portTextField);

// Connect Button

        layout.putConstraint(SpringLayout.WEST, connect, 5, SpringLayout.EAST, portLabel);

        layout.putConstraint(SpringLayout.NORTH,    connect,    5,    SpringLayout.SOUTH,
passwordTextField);


// Boundries

        layout.putConstraint(SpringLayout.EAST, contentPane, 5, SpringLayout.EAST, portTextField);

        layout.putConstraint(SpringLayout.SOUTH, contentPane, 5, SpringLayout.SOUTH, connect);

        frame.pack();

        frame.setVisible(true);

        frame.setLocationRelativeTo(null);

```

```

frame.setDefaultCloseOperation(EXIT_ON_CLOSE);

connect.addActionListener(
    e -> ConnectToServer(nameTextField, ipTextField, portTextField, passwordTextField,
frame));

passwordTextField.addActionListener(
    e -> ConnectToServer(nameTextField, ipTextField, portTextField, passwordTextField,
frame));

}

private static void ConnectToServer(JTextField nameTextField, JTextField ipTextField, JTextField
portTextField,
    JPasswordField passwordTextField, JFrame frame) {
    if (nameTextField.getText().toString().isEmpty() || ipTextField.getText().toString().isEmpty()
        || new String(passwordTextField.getPassword()).isEmpty()
        || portTextField.getText().toString().isEmpty()) {
        String tPass = ((new String(passwordTextField.getPassword())).isEmpty()) ? " Password Field"
: "";
        String tName = (nameTextField.getText().toString().isEmpty()) ? "Name Field" : "";
        JOptionPane.showMessageDialog(null, tName + tPass + " cannot be Empty", "Note",
            JOptionPane.INFORMATION_MESSAGE);

    } else {
        // System.out.println("Vrtified ...");

        CURRENT_USER = nameTextField.getText().toString();
    }
}

```

```

        IP_ADDRESS_STRING = ipTextField.getText().toString();
        PORT = Integer.parseInt(portTextField.getText().toString());
        PASSWORD = new String(passwordTextField.getPassword());

        Client.isSetupDone = true;

        frame.dispose();
    }
}

```

```

Client() {
    super("Chat Window:Client");
    setLayout(new BorderLayout());
    setUI();
    setSize(400, 550);
    setVisible(true);
    setDefaultCloseOperation(3);

    listeners();
}

```

```

private void listeners() {
    send.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            try {

```

```

        if (msg.getText() == null || msg.getText().toString().trim().length() == 0) {

        } else {

            String content = msg.getText().toString();

            msg.setText("");

            DataOutputStream dout = new DataOutputStream(clientSocket.getOutputStream());

            dout.writeUTF(Client.CURRENT_USER + ":::" + Client.enc.encrypt(content,
Client.PASSWORD));

        }

    } catch (Exception e1) {

        e1.printStackTrace();

    }

}

});

msg.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        try {

            if (msg.getText() == null || msg.getText().toString().trim().length() == 0) {

            } else {

                String content = msg.getText().toString();

                msg.setText("");

                DataOutputStream dout = new DataOutputStream(clientSocket.getOutputStream());

                dout.writeUTF(Client.CURRENT_USER + ":::" + Client.enc.encrypt(content,
Client.PASSWORD));

            }

        }

    }

});

```



```

        } catch (Exception e1) {

            e1.printStackTrace();

        }

    }

});

addWindowListener(new WindowAdapter() {

    public void windowClosing(WindowEvent we) {

        try {

            DataOutputStream dout = new DataOutputStream(clientSocket.getOutputStream()); //
sendign      dout.writeUTF("GRP_INFO" + ":::" + Client.CURRENT_USER + " left the Chat.");

            dout.writeUTF("END");

            ObjectOutputStream oout = new ObjectOutputStream(videoSocket.getOutputStream());

            oout.writeObject(new ImageIcon("images\\endImage.png", "END"));

        } catch (Exception e) {

            System.out.println(e);

        }

    }

});

fileSend.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent ae) {

        try {

            jfc.showOpenDialog(null);


```

```

        if (jfc.getSelectedFile() != null) {

            File file = jfc.getSelectedFile();

            FileInputStream fis = new FileInputStream(file.getPath());

            int fileLen = (int) file.length();

            String transferINFO = "FILE_TRANS:::" + file.getName() + ":::" + fileLen + ":::"
                + Client.CURRENT_USER;

            DataOutputStream dos = new DataOutputStream(clientSocket.getOutputStream());

            dos.writeUTF(transferINFO);

            byte b[] = new byte[fileLen];

            fis.read(b, 0, b.length);

            fis.close();

            dos.write(b, 0, b.length);

            dos.flush();

            addMessages("GRP_INFO", "You Send A File");

        }

    } catch (Exception e) {

        e.printStackTrace();

    }

}

});

videoStream.addActionListener(new ActionListener() {

```

```

public void actionPerformed(ActionEvent e) {

    Webcam cam = Webcam.getDefault();

    Client.runCam = true;

    cam.setViewSize(new Dimension(Client.VIDEO_HEIGHT, Client.VIDEO_WIDTH));

    try {

        ImageIcon ic = null;

        BufferedImage br = null;

        ObjectOutputStream vstream = new
ObjectOutputStream(Client.videoSocket.getOutputStream());

        cam.open();

        new VideoOutstreamThread(ic, br, vstream, cam).start();

        new AudioOutStreamThread().start();


    } catch (Exception exception) {

        exception.printStackTrace();

    }

    videoStreamStopUI();

}

});

}

void videoStreamStopUI() {

    JFrame stopFrame = new JFrame();

```

```

stopFrame.setTitle("Pack()");

stopFrame.setLayout(new FlowLayout());

JButton stopButton = new JButton("Stop");

stopFrame.add(stopButton);

stopFrame.pack(); // calling the pack() method

stopFrame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

stopFrame.setLocationRelativeTo(null);

stopFrame.setVisible(true);

stopButton.addActionListener(ae -> {

    Client.runCam = false;

    stopFrame.dispose();

});

stopFrame.addWindowListener(new WindowAdapter() {

    public void windowClosing(WindowEvent we) {

        Client.runCam = false;

        stopFrame.dispose();

    }

});

}

private void setUI() {

    // initila UI setup

    groupName = new JLabel("Connecting...");

```

```

send = new JButton();

fileSend = new JButton();

videoStream = new JButton();

videoStream.setIcon(new ImageIcon("images\\videoicon.png"));

send.setIcon(new ImageIcon("images\\sendicon.png"));

fileSend.setIcon(new ImageIcon("images\\ftpicon.png"));

fileSend.setToolTipText("File Transfer");

videoStream.setToolTipText("Video Stream");

send.setToolTipText("Send");

msg = new JTextField(25);

chat = new JPanel();

scrollPane = new JScrollPane(chat);

jfc = new JFileChooser();


// NORTH

JPanel top = new JPanel();

top.setLayout(new FlowLayout(FlowLayout.CENTER));

add(top, BorderLayout.NORTH);

top.add(groupName);


// CENTER

add(scrollPane, BorderLayout.CENTER);

// chat.setLayout(new BoxLayout(chat , BoxLayout.Y_AXIS));

```

```

// scrollPane.setBorder(new EmptyBorder(10, 10, 10, 10));

chat.setLayout(new BorderLayout());


// SOUTH

JPanel p1 = new JPanel(new BorderLayout());

JPanel p2 = new JPanel(new BorderLayout());

JPanel p3 = new JPanel(new BorderLayout());

add(p1, BorderLayout.SOUTH);

p1.add(p2, BorderLayout.CENTER);

p1.add(send, BorderLayout.EAST);

p1.setBorder(new EmptyBorder(10, 10, 10, 10));

p2.add(p3, BorderLayout.CENTER);

p2.add(fileSend, BorderLayout.EAST);

p3.add(msg, BorderLayout.CENTER);

p3.add(videoStream, BorderLayout.EAST);

}


private void handleFileTransfer(String fileName, String fileLen, String sender, DataInputStream din)
{
    try {
        File directory = new File("FTP Recieved");

        if (!directory.exists())

            directory.mkdir();

        int len = Integer.parseInt(fileLen);

```

```

        FileOutputStream fout = new FileOutputStream("FTP Recieved\\" + fileName);

        byte bytes[] = new byte[len];

        din.readFully(bytes, 0, bytes.length);

        fout.write(bytes, 0, bytes.length);

        fout.flush();

        fout.close();

        addMessages("GRP_INFO", fileName + " recieved from " + sender);

    } catch (Exception e) {

        e.printStackTrace();

    }

}

```

```

private void addMessages(String user, String msg) {

    // Adds Msg in panel Format to add to a chat window

    Color textColor, bgColor;

    FlowLayout layout = new FlowLayout();

    JPanel row = new JPanel();

    JLabel content = new JLabel(msg);

    JLabel sender = new JLabel(user + " ");

    JLabel time = new JLabel(getTime()); // Change to Actual Time

    JPanel message = new RoundedPanel();

```

```

if (user.equals("GRP_INFO")) {

    time.setVisible(false);

    sender.setVisible(false);

    layout.setAlignment(FlowLayout.CENTER);

    textColor = new Color(255, 255, 255);

    bgColor = new Color(110, 103, 103);

} else if (user.equals(Client.CURRENT_USER)) {

    layout.setAlignment(FlowLayout.RIGHT);

    textColor = new Color(255, 255, 255);

    bgColor = new Color(0, 132, 255);

} else {

    layout.setAlignment(FlowLayout.LEFT);

    textColor = new Color(0, 0, 0);

    bgColor = new Color(197, 197, 197);

}

row.setLayout(layout);

message.setLayout(new BoxLayout(message, BoxLayout.Y_AXIS));

sender.setFont(new Font("Helvetica", Font.BOLD, 11));

content.setFont(new Font("Helvetica", Font.PLAIN, 12));

time.setFont(new Font("Helvetica", Font.PLAIN, 10));

message.setBorder(new EmptyBorder(10, 10, 10, 10));

message.setBackground(bgColor);

```



```
sender.setForeground(textColor);

content.setForeground(textColor);

time.setForeground(textColor);


message.add(sender);

message.add(content);

message.add(time);

row.add(message);

chat.add(row, BorderLayout.NORTH); // Adds msg to chat layout

// chat.revalidate();
```

```
JPanel newChat = new JPanel();

newChat.setLayout(new BorderLayout());

chat.add(newChat, BorderLayout.CENTER);

chat = newChat;

chat.revalidate();
```

```
JScrollBar vertical = scrollPane.getVerticalScrollBar();

vertical.setValue(vertical.getMaximum());

JScrollBar vertica = scrollPane.getVerticalScrollBar();

vertical.setValue(vertica.getMaximum());
```

```
}
```

```

private String getTime() {

    Date date = new Date();

    SimpleDateFormat formatter = new SimpleDateFormat("hh:mm a");

    return formatter.format(date);

}


public static void main(String[] args) {

    // System.out.println("Start");

    while (!Client.isSetupDone) {

        System.out.print("");

    }

    // Wait till u get all info

    Client client = new Client();

    try {

        client.clientSocket = new Socket(IP_ADDRESS_STRING, PORT);

        DataInputStream din = new DataInputStream(client.clientSocket.getInputStream());

        String groupName = din.readUTF();

        client.groupName.setText(groupName);

        DataOutputStream dout = new DataOutputStream(client.clientSocket.getOutputStream());

        // Verification

```

```

String request = din.readUTF();

if (request.startsWith("RequestSecretText")) {

    dout.writeUTF(enc.encrypt(Client.PASSWORD, Client.PASSWORD));

} else {

    try {

        String str = dec.decrypt(request, Client.PASSWORD);

        if (!str.equals(Client.PASSWORD)) {

            JOptionPane.showMessageDialog(client, "You Have entred Wrong Password", "Invalid
Password",

                JOptionPane.ERROR_MESSAGE);

            System.exit(0);

        }

    } catch (IllegalBlockSizeException | BadPaddingException e) {

        client.dispose();

        JOptionPane.showMessageDialog(client, "You Have entred Wrong Password", "Invalid
Password",

            JOptionPane.ERROR_MESSAGE);

        System.exit(0);

    } catch (Exception e) {

        e.printStackTrace();

    }

}

new ClientVideoStreamThread().start();

new ClientAudioStreamThread().start();

```

```

dout.writeUTF("GRP_INFO" + ":::" + Client.CURRENT_USER + " joined the Chat.");

while (true) {

    String response = din.readUTF();

    String[] str = response.split(":::");

    if (str[0].equals("FILE_TRANS")) {

        client.handleFileTransfer(str[1], str[2], str[3], din);

    } else if (str[0].equals("GRP_INFO"))

        client.addMessages(str[0], str[1]);

    else

        client.addMessages(str[0], Client.dec.decrypt(str[1], Client.PASSWORD));

}

} catch (java.net.ConnectException e) {

    client.groupName.setText("FAILED !");

    JOptionPane.showMessageDialog(client, "Server doesn't exist : Invalid IP Address", "Server
Not Found",

        JOptionPane.ERROR_MESSAGE);

    System.exit(0);

} catch (java.io.EOFException e) {

    System.out.println("Ended");

} catch (Exception e) {

    e.printStackTrace();

}

}

}

```

```

class ClientVideoStreamThread extends Thread {

    Socket videoSocket;

    public void run() {

        try {

            videoSocket = new Socket(Client.IP_ADDRESS_STRING, Client.PORT + 1);

            Client.videoSocket = videoSocket;

            JFrame videoFrame = Client.videoFrame;

            ImageIcon ic;

            JLabel videoFeed = new JLabel();

            // videoFrame.setLayout(null);

            videoFrame.setTitle("Client :" + Client.CURRENT_USER);

            videoFrame.add(videoFeed);

            videoFrame.setVisible(false);

            videoFrame.setSize(Client.VIDEO_HEIGHT, Client.VIDEO_WIDTH);

            videoFrame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

            videoFrame.addWindowListener(new WindowAdapter() {

                public void windowClosing(WindowEvent e) {

                    videoFrame.setVisible(false);

                }

            });

            while (true) {

```

```

        ObjectInputStream oin = new ObjectInputStream((videoSocket.getInputStream()));

        ic = (ImageIcon) oin.readObject();

        videoFeed.setIcon(ic);

        if (!videoFrame.isVisible())

            videoFrame.setVisible(true);

        if (ic != null && ic.getDescription() != null && ic.getDescription().equals("END_VIDEO"))
    {

        videoFrame.setVisible(false);

    }

}

    } catch (java.io.EOFException e) {

        System.out.println("Ended");

    } catch (Exception e) {

        e.printStackTrace();

    }

}

}

```

```

class VideoOutstreamThread extends Thread {

    ImageIcon ic;

    BufferedImage br;

    ObjectOutputStream stream;

    Webcam cam;

```

```

VideoOutstreamThread(ImageIcon ic, BufferedImage br, ObjectOutputStream stream, Webcam
cam) {

    this.ic = ic;

    this.br = br;

    this.stream = stream;

    this.cam = cam;

}

public void run() {

    try {

        while (Client.runCam) {

            br = cam.getImage();

            ic = new ImageIcon(br);

            stream.writeObject(ic);

            stream.flush();

        }

        ic = new ImageIcon("images\\endVideo.png", "END_VIDEO");

        stream.writeObject(ic);

        stream.flush();

    } catch (Exception e) {

        e.printStackTrace();

    }

    cam.close();

}

```

```
}
```

```
class ClientAudioStreamThread extends Thread {  
  
    Socket audioSocket;  
  
    ObjectInputStream ois;  
  
    AudioFormat format;  
  
    DataLine.Info info;  
  
    SourceDataLine speakers;  
  
    byte[] data;  
  
    public void run() {  
  
        try {  
  
            audioSocket = new Socket(Client.IP_ADDRESS_STRING, Client.PORT + 2);  
  
            Client.audioSocket = audioSocket;  
  
            data = new byte[1024];  
  
            format = new AudioFormat(48000.0f, 16, 2, true, false);  
  
            info = new DataLine.Info(SourceDataLine.class, format);  
  
            data = new byte[1024];  
  
  
            speakers = (SourceDataLine) AudioSystem.getLine(info);  
  
            speakers.open(format);  
  
            speakers.start();  
  
            ois = new ObjectInputStream(audioSocket.getInputStream());  
  
            while (true) {
```



```

        int dsize = ois.read(data);

        if (dsize == 1024) {

            speakers.write(data, 0, dsize);

        }

    }

    } catch (Exception e) {

        e.printStackTrace();

    }

}

```

```

class AudioOutputStreamThread extends Thread {

```

```

    private ObjectOutputStream oos;

    private AudioFormat format;

    private DataLine.Info info;

    private TargetDataLine microphone;

    private byte[] data;

    private int dsize;

```

```

    AudioOutputStreamThread() {

```

```

    }

```

```

    public void run() {

```

```

try {

    // Audio Stuff

    format = new AudioFormat(48000.0f, 16, 2, true, false);

    microphone = AudioSystem.getTargetDataLine(format);

    info = new DataLine.Info(TargetDataLine.class, format);

    data = new byte[1024];


    microphone = (TargetDataLine) AudioSystem.getLine(info);

    microphone.open(format);

    microphone.start();

    oos = new ObjectOutputStream(Client.audioSocket.getOutputStream());

    // read and send part

    while (Client.runCam) {

        dsize = microphone.read(data, 0, data.length);

        oos.write(data, 0, dsize);

        oos.reset();

    }

    System.out.println("[ Client ] : Attempting to stop ");

    oos.write(data, 0, 512);

    oos.flush();

} catch (Exception e) {

    e.printStackTrace();

}

```

```

        microphone.stop();

        microphone.close();
    }
}

```

2.3 SOURCE CODE FOR ENCRYPTION:

```

import java.nio.charset.StandardCharsets;

import java.security.AlgorithmParameters;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class Encryption {

    public String encrypt(String word, String password) throws Exception {

        byte[] ivBytes;

        SecureRandom random = new SecureRandom();

        byte[] bytes = new byte[20];

        random.nextBytes(bytes);

        byte[] saltBytes = bytes;

        // Derive the key

```

```

    SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");

    PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), saltBytes, 1500, 256);

    SecretKey secretKey = factory.generateSecret(spec);

    SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(), "AES");

    //encrypting the word

    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

    cipher.init(Cipher.ENCRYPT_MODE, secret);

    AlgorithmParameters params = cipher.getParameters();

    ivBytes = params.getParameterSpec(IvParameterSpec.class).getIV();

    byte[] encryptedTextBytes = cipher.doFinal(word.getBytes(StandardCharsets.UTF_8));

    //prepend salt and vi

    byte[] buffer = new byte[saltBytes.length + ivBytes.length + encryptedTextBytes.length];

    System.arraycopy(saltBytes, 0, buffer, 0, saltBytes.length);

    System.arraycopy(ivBytes, 0, buffer, saltBytes.length, ivBytes.length);

    System.arraycopy(encryptedTextBytes, 0, buffer, saltBytes.length + ivBytes.length,
encryptedTextBytes.length);

    return new Base64().encodeToString(buffer);

}
}

```

2.4 SOURCE CODE FOR DECRYPTION:

```

import java.nio.ByteBuffer;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

```

```

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;


public class Decryption {


    public String decrypt(String encryptedText, String password) throws Exception {

        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

        //strip off the salt and iv

        ByteBuffer buffer = ByteBuffer.wrap(new Base64().decode(encryptedText));

        byte[] saltBytes = new byte[20];

        buffer.get(saltBytes, 0, saltBytes.length);

        byte[] ivBytes1 = new byte[cipher.getBlockSize()];

        buffer.get(ivBytes1, 0, ivBytes1.length);

        byte[] encryptedTextBytes = new byte[buffer.capacity() - saltBytes.length - ivBytes1.length];

        buffer.get(encryptedTextBytes);

        // Deriving the key

        SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");

        PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), saltBytes, 1500, 256);//65556
    }
}

```

```

        SecretKey secretKey = factory.generateSecret(spec);

        SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(), "AES");

        cipher.init(Cipher.DECRYPT_MODE, secret, new IvParameterSpec(ivBytes1));

        byte[] decryptedTextBytes = null;

        try {

            decryptedTextBytes = cipher.doFinal(encryptedTextBytes);

        } catch (IllegalBlockSizeException e) {

            throw e;

        } catch (BadPaddingException e) {

            throw e;

        }

        return new String(decryptedTextBytes);

    }
}

```

2.5 SOURCE CODE FOR ROUNDEDPANEL:

```

import javax.swing.*;

import java.awt.*;

public class RoundedPanel extends JPanel {

    RoundedPanel() {

        super();

        setOpaque(false);

    }
}

```

```

/**
 *
 */

private static final long serialVersionUID = 1L;

/** Stroke size. it is recommended to set it to 1 for better view */
protected int strokeSize = 1;

/** Color of shadow */
protected Color shadowColor = Color.black;

/** Sets if it drops shadow */
protected boolean shady = true;

/** Sets if it has an High Quality view */
protected boolean highQuality = true;

/** Double values for Horizontal and Vertical radius of corner arcs */
protected Dimension arcs = new Dimension(20, 20);

/** Distance between shadow border and opaque panel border */
protected int shadowGap = 1;

/** The offset of shadow. */
protected int shadowOffset = 1;

/** The transparency value of shadow. ( 0 - 255) */
protected int shadowAlpha = 150;

```

```

@Override

```

```

protected void paintComponent(Graphics g) {

    super.paintComponent(g);

    int width = getWidth();

    int height = getHeight();

    int shadowGap = this.shadowGap;

    Color shadowColorA = new Color(shadowColor.getRed(), shadowColor.getGreen(),
shadowColor.getBlue(),

        shadowAlpha);

    Graphics2D graphics = (Graphics2D) g;

    // Sets antialiasing if HQ.

    if (highQuality) {

        graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

    }

    // Draws shadow borders if any.

    if (shady) {

        graphics.setColor(shadowColorA);

        graphics.fillRoundRect(shadowOffset, // X position

            shadowOffset, // Y position

            width - strokeSize - shadowOffset, // width

            height - strokeSize - shadowOffset, // height

            arcs.width, arcs.height); // arc Dimension

    } else {

```



```

        shadowGap = 1;
    }

    // Draws the rounded opaque panel with borders.

    graphics.setColor(getBackground());

    graphics.fillRoundRect(0, 0, width - shadowGap, height - shadowGap, arcs.width, arcs.height);

    graphics.setColor(getForeground());

    graphics.setStroke(new BasicStroke(strokeSize));

    graphics.drawRoundRect(0, 0, width - shadowGap, height - shadowGap, arcs.width, arcs.height);

    // Sets strokes to default, is better.

    graphics.setStroke(new BasicStroke());
}
}

```

CHAPTER 3

SNAPSHOTS

3.1 SERVER INITIALIZATION

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

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

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\project> & 'C:\Program Files\Java\jdk1.8.0_301\bin\java.exe' '-cp' 'C:\Users\PRAVEE~1\AppData\Local\Temp\cp_3m1xp7chboj09ysgmz8ofz20y.jar' 'S
erver'
Enter The Group Name :Whatsapp
Server Created with Port No: 2000 and Listening ...
█
```

Figure3.1:Server Initialization

3.2 CLIENT SIDE

Client Side 1	Client Side 2
<p>Name : <input type="text" value="Test1"/></p> <p>IP Address : <input type="text" value="localhost"/></p> <p>Port : <input type="text" value="2000"/></p> <p>Password : <input type="password" value="...."/></p> <p><input type="button" value="Connect !"/></p>	<p>Name : <input type="text" value="Test2"/></p> <p>IP Address : <input type="text" value="localhost"/></p> <p>Port : <input type="text" value="2000"/></p> <p>Password : <input type="password" value="...."/></p> <p><input type="button" value="Connect !"/></p>

Figure 3.2:Client Side

3.3 MULTI-CHAT

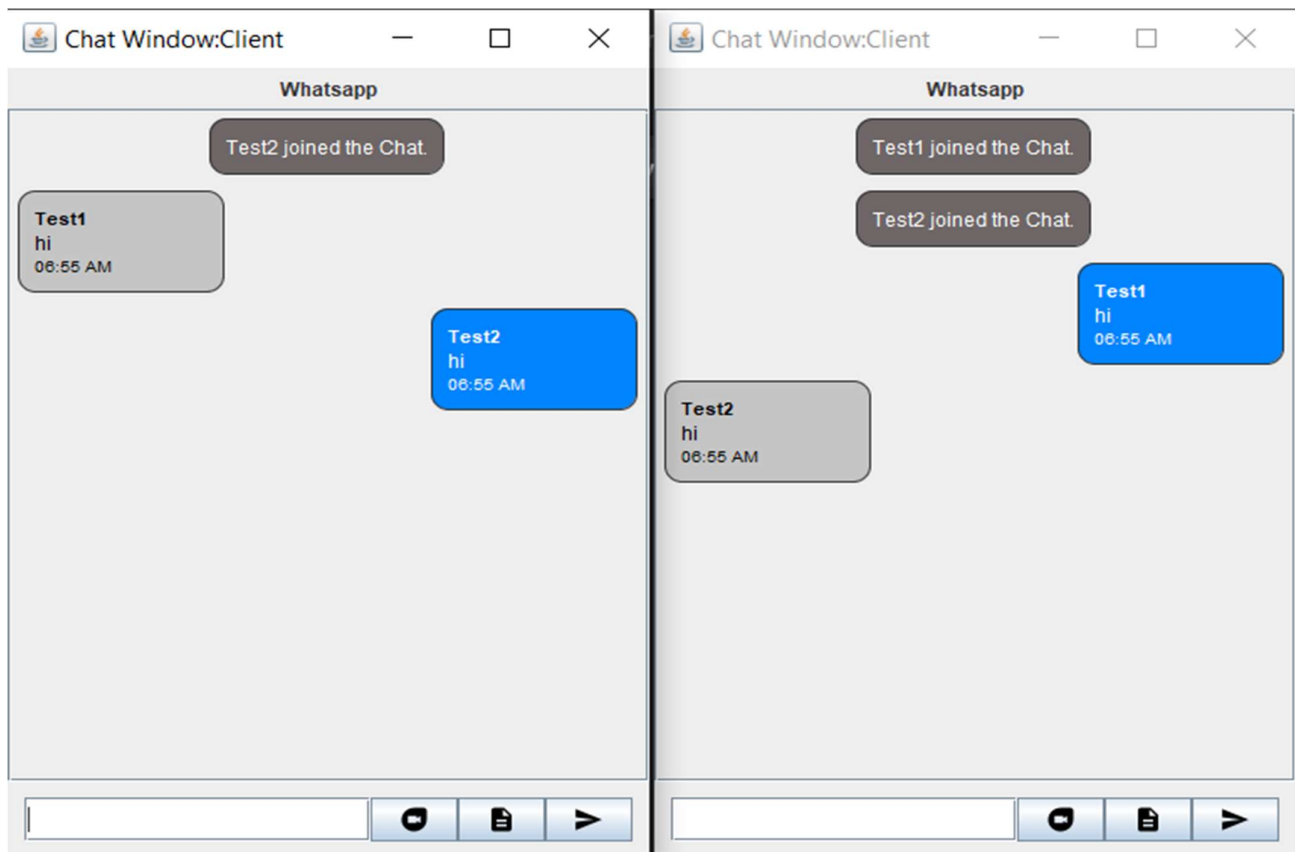


FIGURE 3.3:Multi-Chat

3.4 FILE TRANSFER AND RECEIVING

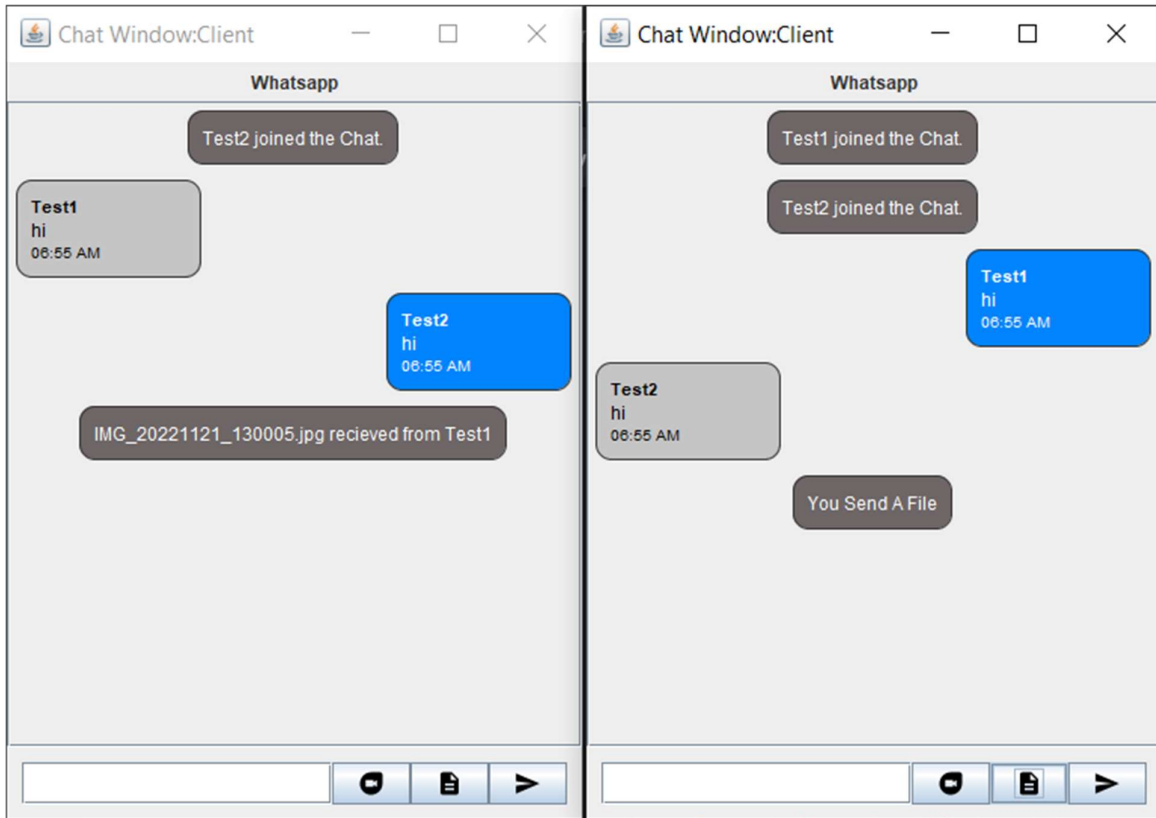


FIGURE3.4: Intimation of File Received

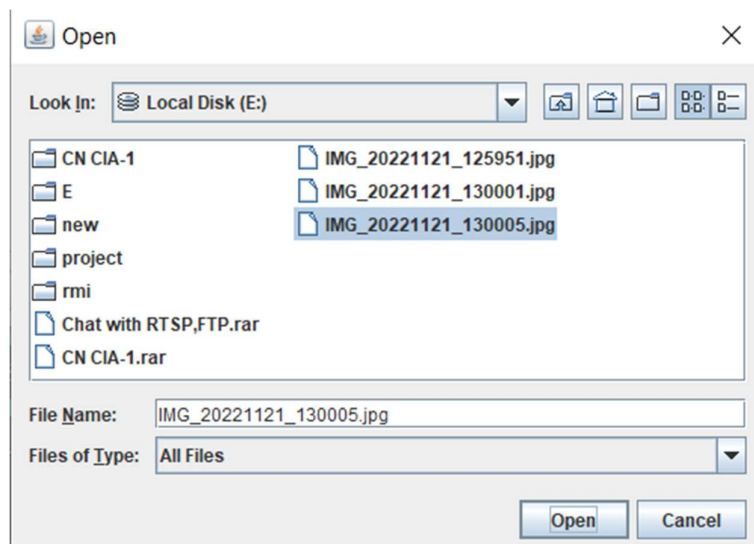


Figure 3.5: File Selection

in the PC

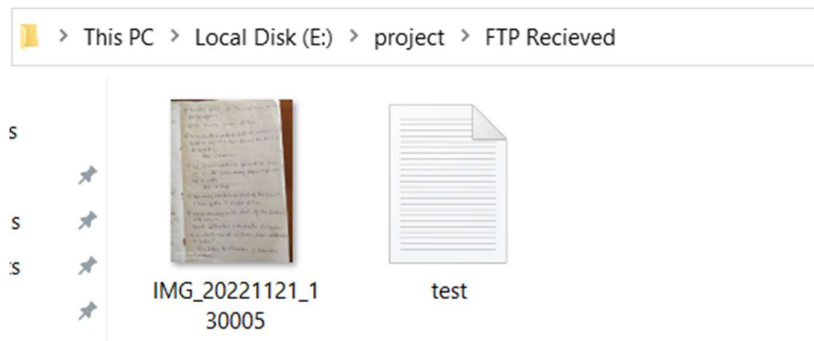


Figure 3.6 : File Received

3.5 Video Streaming

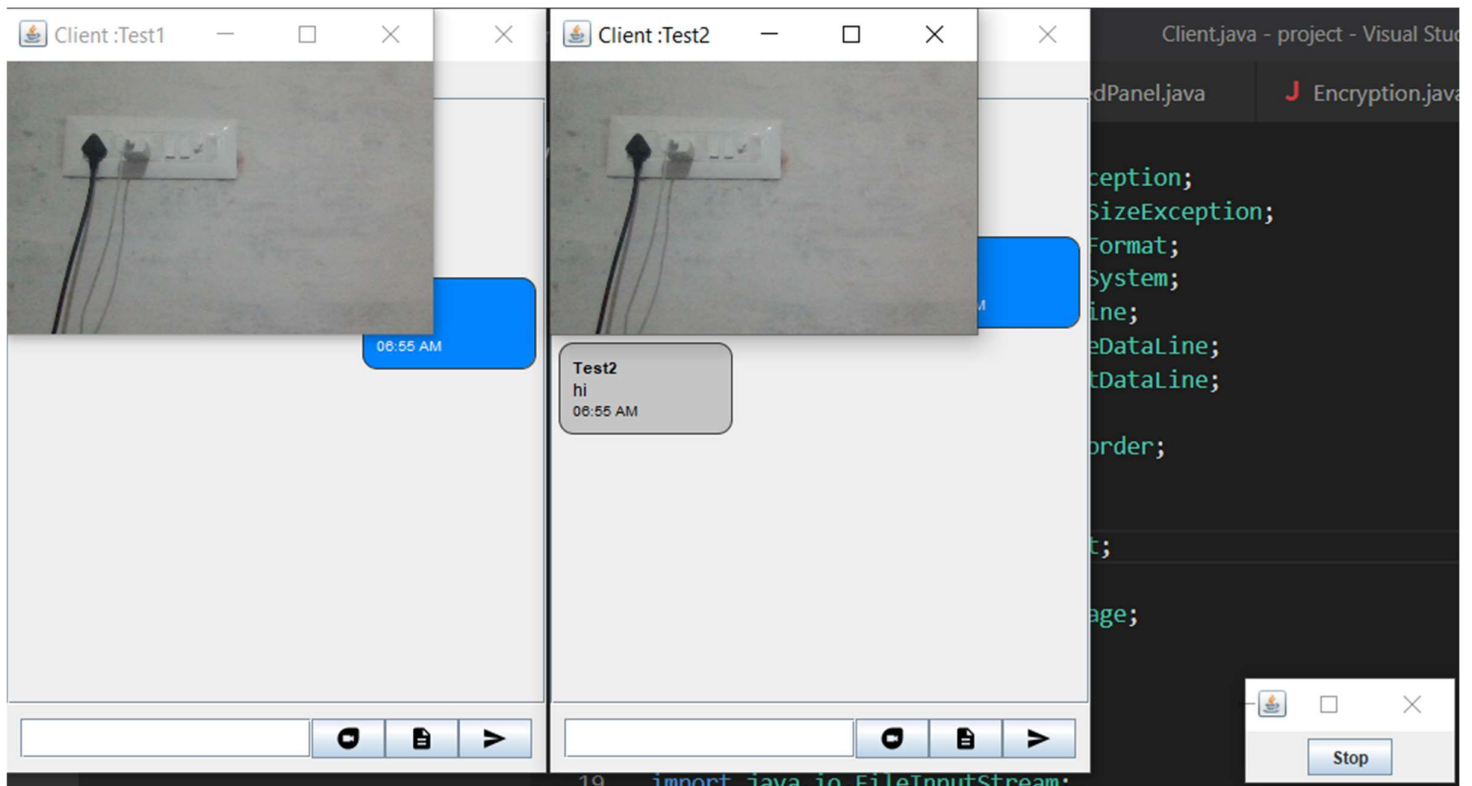


Figure 3.7: Video Streaming

4.CONCLUSION

Our project is only a humble venture to satisfy the needs to manage their project work. Several user-friendly coding have also been adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a framework that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses. The chat app provides a better and more flexible chat system. Developed with the latest technology in the way of providing a reliable system. The main advantage of the system is instant messaging, real-world communication, added security, group chat, etc. This application may find the best demand in the market for most organizations that aim to have independent applications.