

IMPLEMENTATION OF MULTI-USER SECURE CHATROOM WITH ENCRYPTION USING UDP AND SMTP

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

CSE302: COMPUTER NETWORKS

Submitted by

Madireddy Kalyan Venkat

(Reg. No.: 122003141, Computer Science & Engineering)

February 2021



SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401



SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled “**IMPLEMENTATION OF MULTI-USER SECURE CHATROOM WITH ENCRYPTION USING UDP AND SMTP**” submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a Bonafide record of the work done by **Madireddy Kalyan Venkat (Reg. No.: 122003141, Computer Science & Engineering)** during the academic year 2020-21, 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	Datagram Packet format	1
1.2	SMTP flow	2
1.3	Encryption and decryption outline	4
3.1	application run	74
3.2	mail id pane	74
3.3	mail id entry pane	74
3.4	passcode pane	74
3.5	mail screen shot	75
3.6	pass code entry	75
3.7	host or client	75
3.8	host entry pane	76
3.9	host entry filled pane	76
3.10	host application	77
3.11	client entry pane	77
3.12	client application 1	78
3.13	client application 2	78
3.14	client application 3	79
3.15	joined display 1	79
3.16	joined display 2	80
3.17	send message	80
3.18	receive message	81
3.19	clear button ask	82
3.20	clear button result	82
3.21	profile button result	83

3.22	profile button result modified	83
3.23	participants button result	84
3.24	count button result	84
3.25	copy text button result	85
3.26	settings menu	85
3.27	frame colour change 1	86
3.28	frame colour change 2	86
3.29	menu colour change 1	87
3.30	menu colour change 2	87
3.31	font menu	88
3.32	font change	88
3.33	font size change	89
3.34	font colour change	89

List of Tables

Table No.	Table name	Page No.
1.1	Brief resource table	4

Abbreviations

UDP	User Datagram Protocol
SMTP	Simple Mail Transfer Protocol
IP	Internet Protocol
GUI	Graphical User Interface
AES	Advanced Encryption Standard
UA	User Agent
MTA	Mail Transfer Agent
SA	Socket Address

Notations

‘ * ’	All
Name()	Function definition
‘ @ ’	Separates user-id from domain name

Abstract

Implementation of Multi-user Secure Chatroom with Encryption Using UDP and SMTP is an application program that focuses on the usage of User Datagram Protocol, Simple Mail Transfer Protocol and Encryption-Decryption concepts. This application allows the users to create a chat room or to join in an already created chat room.

The User who created a chat room acts as the server or host to that particular chat room. All the remaining users who wants to join in that particular chatroom should verify their mail addresses using the one time pass code they receive through their mail and should enter the IP-address of the host of that chat room to enter into that particular chat room. Here the security of the chat room is maintained through the Mail-Authentication. Any number of users can enter into a chat room and send the messages to all the remaining participants or receive the messages from any participant of the chat room. All the messages circulated through the chat room were protected using the AES-encryption and decryption techniques. The message is encrypted before sending and decrypted while receiving. The messages are sent and received using the simple message-oriented transport layer protocol known as UDP. Here the GUI used in the application is ample and makes the usage of application robust and user friendly.

Key Words: UDP, SMTP, encryption, decryption, Multiple Users

Table of Contents

Title	Page No.
Bonafide Certificate	ii
List of Figures	iii
List of Tables	iv
Abbreviations	v
Notations	vi
Abstract	vii
1. Introduction	1
2. Source code	12
3. Snap shots	74
4. Conclusion and Future plans	90
5. References	91

Introduction

The major focus of this project is around UDP, SMTP and encryption and decryption methods, first let's discuss these terms at brief,

UDP

UDP stands for user datagram protocol. It is communication protocol mainly used to establish the exchange of information in a network of computers or systems. It comes under the Transport layer protocol of network layers and is unreliable and connectionless. It's used for mostly answering the query type communications. It's also called the stateless protocol because of the reason that it doesn't acknowledge the sent data. Low latency and low bandwidth connections can also be favoured by the UDP. UDP when used along IP in a network can sometimes be called UDP/IP suite. Process communication is well preferred by this protocol. Areas of the UDP usage includes-Gaming over networks, Audio and Video chatting applications and so on

UDP uses UDP-headers for packaging the messages to transfer over the network. This header in turn consists of four segments called fields for specific functions. These fields are

1. Source port number
2. Destination port number
3. Length
4. Checksum

Source port number-contains the number of the sender or host

Destination port number-Contains the port of the client or receiver

Length-The length of the UDP header in bytes and other added data

Checksum-its main purpose of usage is error checking

Each of these field is 2 bytes each making the total of 8 Bytes which is the fixed size of the UDP header. Datagram consists of the header and data part. Data part consists of the all other information that will be sent.

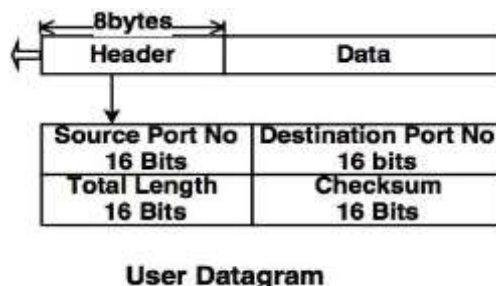


Fig. 1.1.Datagram Packet format.

UDP takes the help of IP to transfer the data from one end to the other end in a network. At first UDP combines the data in to a Datagram packet and adds the other information of the header like Source port from which communication initiated, Destination port to which it's sending, the packet length and the check sum required for the completion of the UDP packet. Later these UDP packets are encapsulated into an IP packets and are sent over the network. It implies simple transmission model while sending the information. UDP is mostly used in the scenarios were the data transmission rate is given at most importance. Since it supports packet switching at ease, it is mostly used in multi user and multi casting applications.

SMTP

One of the important protocols in the application layer of the networking is SMTP, Which stands for Simple Mail Transfer Protocol. As the name suggests it plays an important role in sending the mail's over the network. This protocol uses two modes of communication establishment techniques, the first one is End to end model and the second one being Store and forward type. Both these methods has their own area of preferential usage. In other words the simple exchange of mails between various users is supported by this protocol. It does this by simply defining the set of rules or guidelines which helps in the exchange of the mails. Information sent through mails can be of the form of text or images or video or other file types and can be sent between or more number of users.

It is also based on the client-server architecture as a result leads in the discussion of two components, SMTP server and SMTP client. These were in turn broken in to the two important parts User agent simply known as UA and Mail Transfer Agent simply known as MTA. The functions prepared by UA are-Preparation of the message, Creation of envelop, placing the message in envelop. Whereas the main function of the MTA is to transfer that envelop mail across the network to the respective user. MTA maintains a queue for sending the messages in-case of failure to resend them. The user of the SMTP only deals with the UA, the MTA which delivers the mails to the other user's mail boxes abstracts its procedure from the user. The well-known examples of the UA are Microsoft outlook, Mozilla.

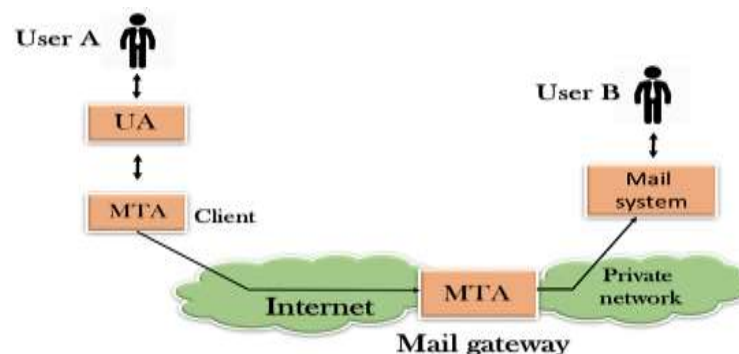


Fig. 1.2.SMTP flow.

At first the mail is composed using the User agent, the mail here consists of two parts body and the header, where the body consists of the information in the mail and header part consists of the sender and receiver addresses along with the accumulate information such as subject of the mail. After this enveloping procedure, the mail is submitted to the SMTP server by the mail client. The mail id is used for the delivery of the right mail to the right user, the mail id consists of the user name and domain name which helps in the successful sending of the message, if the domains of the client and server are different then exchange server comes in to action. After receiving the incoming message the exchange server delivers it to the Mail delivery agent where it is stored for the user to retrieve it. This retrieval is done by the Mail User Agent which is in turn protected by the password.

Encryption and Decryption

Cryptography serves as the means of protecting data through various mathematical functions. The key terms used in this are plaintext which is the original message, cipher text which is the output of the encryption technique, Key which when given along the plain text converts in to the cipher. JCA is the Java Cryptography architecture that helps in the cryptography process in the java programming.

Encryption is the process of converting the plain text into the cipher text given the key, the reading of cipher text will not be possible unless otherwise you have the key. This encryption is done to maintain the confidentiality, Integrity, Authentication and non-repudiation. Encryption can be of both symmetric and Asymmetric. In Symmetric Encryption both the sender and the receiver uses the same key where as in the non-symmetric version both uses two different keys that were mathematically related. There are two prominent classes used to achieve this process they are Secure Random class and Key generator class. The secure random class helps in generating the secure random number which we use for encryption and decryption techniques. Key generator class generates the key with certain algorithms such as AES, DES and so on.

Similarly in the decryption the cipher text received in the receiver side is not in the understandable format which is then decrypted using the Key available with the receiver, either it may be a shared key or the private key that is mathematically related to the public key of the sender. Once the cipher text is decrypted using the key, it changes into the readable format that is plain text. This is how the encryption and decryption of the Messages are done to maintain the safety of data Privacy and integrity.

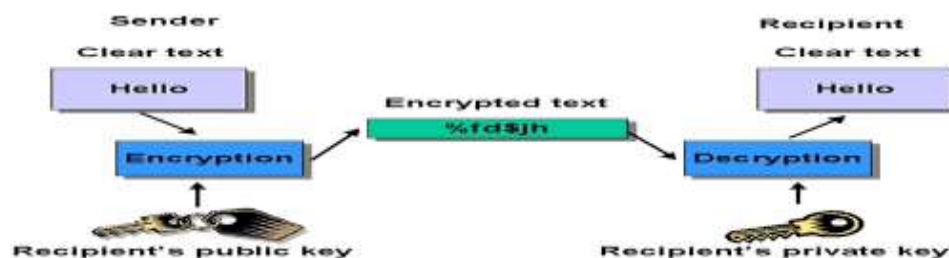


Fig. 1.3.Encryption and decryption outline.

Table 1.1 Brief resource table.

Mail server used	SMTP
Cryptography algorithm	AES
Protocol	UDP

DETAILED-EXPLANATION

This Application project is based on the traditional server-client mechanism where there can be ‘n’ number of clients but a single server, in other terms this can also be called as request-granted mechanism. We will categorize this into two parts host and client and explain them separately. But before that we will see the entry part into the application which focuses on the authentication through mail or SMTP.

Authentication through mail

On first running of the application, you will be asked to provide your mail-id, once you provide the mail-id in the given area, you will be asked to enter the pass-code received through the mail to enter in to the application. Entry will be granted if and only if the entered passcode is correct. The passcode is similar to the OTP except for the fact that this passcode contains the 6 digit randomly generated string which includes all the capital, small alphabets and numbers. This is done to make sure that Authentication of one-self is done in a genuine manner. This passcode sent to the user who successfully runs the application is done by using the Simple Mail Transfer Protocol. The SMTP program is stored in the user defined package *mailcode* in the name of *mailsend.java*. The main Program imports the package specified and creates an object that calls the function *sendmail()* which takes the user-given mail id as the parameter and assigns it to the receiver String to whom the application is sending mail. The *mailsend.java* program in turn consists of a user-defined package *generatornumber* which implements the program *randomnumbergenerator.java* which returns a randomly generated string of length 6. This returned String is stored and sent as the body of the mail with added message attributes like from, to and subject as “login confirmation”. The mail attribute-from is defined in the program is constant. Here we are using the smtp.gmail.com as the host mail property. On the success of the sent message the *sendmail()* function called in the main application also returns the same passcode that is generated and sent. This returned string is stored and compared against the passcode entered by the user in the next step. If the entered Passcode matches with the stored passcode then the application proceeds further, if not then the program notifies the user about the wrong passcode and prevents the user from entering in to the application. The detailed image presentation seen in the chapter 3 of the snapshot part of this Report. The next continuation is

Deciding Host or Client

The next step after correct authentication is deciding on host or client where the user will be given the option of choosing between the host and client. If the user decides to act as the host, then he needs to select the ok button for host if not for the host then he can become the client. If some other user on the same IP address already selected to be a host, then the other users after that cannot become the host, however one can become the client after that. After that you have to enter certain details with respect to the option chosen to proceed further into the application.

Entry of details

If you did select the yes to host then you will get a white background frame consisting of two fields to be filled by you. One is your name, where you need to type your name, the other field is Chatroom name field where you will be given the option of naming your chat room as you are the host. The entered chatroom name will serve as the basis for all the other users who wants to join in that particular chatroom.

If you did select the yes to client then you will be directed to a white background frame consisting of two fields to be filled by you. One is the name field where one has to enter their name i.e user name. The other field is the host-IP address field where one has to enter the IP-address of the host who created the chatroom to make sure they enter in to that room. If you give the wrong IP-address of the chatroom then you will be notified to correct the IP-address.

This is where the application breaks in to the two parts that does their own actions and perform functions apart from other. The two parts are host mode and client mode parts. Which opens on the click of OK button after the entry of the details respectfully.

HOST MODE-PART

The filled fields of name and chatroom name are taken as the parameters of the class constructor to invoke the object. The object creation requires 4 parameters, 2 of which are name and room name, the other 2 are IP address and mode, the mode is what that decides the host or client creation, here in this application if the mode is zero then host or else if the mode is 1 then it is client, so we are giving 0 here as the 3rd parameter. The 4th parameter IP address is given null because the host has its own IP which will be assigned later. All these given parameters are stored in to the variables declared for the class. The GUI of this application is explained later in this chapter. But for now we will use the text field for typing message and text area for showing the whole history of sent and received messages. The text you want to send should be typed in the text field. Later that text is extracted and will be added the name in front of it making it a new string which looks like this *Name: "message you typed"*, this string is further encrypted using the imported package which will be discussed later in this chapter. After the encryption part if string

length is not null then we will invoke a function called *broadcast(String s)* which takes the string as the parameter, here in this case the string is the encrypted form of the name and typed text in the text field of the host mode. In this host mode the socket with port number 37988 is created to send the messages.

What happens in the function *broadcast()*?

Whenever this function is called a datagram packet is created with the parameters of string length, string bytes. Here to note that whenever any user joins the chat room as the client, their details will be saved in the array list called client list, this part will be explained later under the client mode heading in this chapter. This client list consists of the IP address, Port number of all the clients in that chatroom at that point of time. For the created packet say pack which still requires 2 more parameters address and port number, is given in the form of the for loop starting from the first client to the last client and in parallel sending the packet with the help of the socket created in the host mode previously. "In simple words the encrypted string which is given as a parameter is sent to all the clients in the chatroom." There is a decryption function in the one of the user defined packages to decrypt the string which will be discussed later in this chapter. The encrypted string which was sent to all clients is taken and trimmed to remove all the extra bytes and decrypted using the package function and added to the Text Area of the host side. There will be also a thread which will be running all the time, what it does is that it creates a new package and starts listening/waiting for the data sent by the client. Once the data is available it checks whether the string contains "!!^^" part or not, if it contains then that means the received string is of the form "!!^^"+Name+"^^!!" where the name is the name of the client joined. It extracts the Name from that string and adds it to the client list name part, it also finds out the IP address and port number of the client with the help of the received packet and adds that information into the client list. This is how the host gets to know the client's data. After that the received name of the client is converted in to the format "Name+ joined the chatroom" and encrypted into the string and invokes *broadcast* for that string. What happens in invoking the function *broadcast* is discussed above. Not only this it also sends the chatroom name in the form of "!!^^"+chatroom name+"^^!!" to the client created to make sure that the chatroom name is shared with every client in the chatroom name. If the received string data by the host does not contain the "!!^^" part that means the client is not new and some message typed by the user is received then it just broadcasts the received message. This process syncs with the client mode part in implementation.

Client Mode-Part

Similar to the host mode, the client mode part is created with the help of the parameters which are given at the beginning of the application. The object is called with the parameters with the parameters name, IP address, mode which is 1 here in the client side part, and the chatroom name which will initially be empty or null because the client do not know the name of the chat room, he will get to know the name of the chatroom only after the entry into the room application. After the creation of the client side, similar to

the host side, it will have both text field and text area. As in the case of the host, here also it checks the encrypted string of name and message typed of the form “name: message typed”. If the string contains at least one character then it invokes the function *sendtohost(String s)* which we will discuss. But before that there are some other activities that will be performed in the client side on the entry they are: new datagram socket will be created with no port number. The next one is that we define a string with the name given initially as the parameter, in this form “!!^^”+Name+“^^!!” Then we define a packet with the parameters of the string above converted into bytes, length of the string, host IP, 37988 to send the string to the host. After sending the string above to the host, the client creates other packet and starts listening/waiting for the sent data from the host. Once the data is received from the host, it checks whether the string received contains the “!!^^” part in it or not. If it contains this part then it understands that the received data is not the message typed by the host but it is the chatroom name sent by the host in the form of string “!!^^+chatroom name+^^!!” this received chat room name in the different format is extracted separately out and is set as visible for of the application using the java Label on the top right corner of the application, this is how the client gets to know the name of the chat room after he joins. The client mode similar to the host mode also runs the thread which will be always listening for some data with some packet. Once the data is available it will be decrypted and added to the text area part of the client. Now we will see what happens when we call *sendtohost()* function . It just sends the string in the parameter to the host through the creation of new packet. This is what happens in client mode. This client mode works in sync with the host mode to make the application running all along.

Simple Words

In simple words, at first the host is created and a finite number of clients are created then every created client sends the name of its user to the host, in response the host receives the name of the user client from every client that joins the chatroom and saves their IP addresses and port numbers along with the client user names in the array list called client list with the help of received packets. In also response to that, the host sends its chatroom name to all the clients created. This is how the host and clients gets connected. Later when the host wants to send some message in the chat room, it will encrypt the message and will call the broadcast function, which simply does two things. The first one is it sends that encrypted message to all the clients in the chatroom using the details in the client list. The second thing that this function does is that it simply decrypts the string and copies it to the text area part of the host side. The client which will be always waiting for the data to be received, on receiving the string of data simply decrypts the string and adds it to the text area part of the respective client. This is what happens when the host sends the message in the chatroom, when any of the client wants to send the message it simply encrypts the string and calls the send to host function which simply sends the string to the host side. The host again calls the send to host function for this received string to make sure it reaches the text areas of every client in the chatroom. This is how the messages can be shared between the ‘n’ number of clients and host. Now we will see the encryption-

decryption implementation part that is how the message is encrypted before sending and how it is decrypted after receiving.

How Encryption?

The code required to do this is stored in the package called my pack and the object for the class AES Program is created to perform this operation. The class consists of mainly two functions encrypt() and decrypt() for performing the required operations. Both these functions return the corresponding string in accordance with their names. At first user calls the encryption function with two String parameters one is the key of the string, which user specifies to do the necessary operations. Note this key should also be present along with the decryption performing application if it has to do the decryption or else the simple text cannot be generated from the encrypted cipher text. The other parameter that is passed to the encryption function is plain text string or original message string. After invoking the encrypt function it further invokes the setkey() function which takes the parameter string key, what setkey function does is that first it changes the given string key into UTF bytes from UTF bytes it changes the arbitrary byte array into the fixed length hashes using message digest instance and then creates a new secret key spec instance key using the algorithm “AES” and stores it into the secret key. After this function completes execution the next step is executed. The cipher object is created using the getInstance method with the algorithm “AES” mode “CBC” and padding “PKCS5Padding”. Then cipher is initialized with the key generated in the name of secret key with mode as Encrypt_mode. Then base64 encoder is used to encode the plain text using the initialized cipher. And this encoded string is returned to the place of call. Which is circulated through the network for protection of data.

How Decryption?

Decryption also starts with calling the setkey method to change the key string passed as a parameter into the “AES” secret key through series of steps using the message digest class. After that similar to the encryption the cipher object is created with the getInstance method. After that the cipher object is initialized using the init method with two parameters secretkey and the Decrypt_mode adding the real spice of difference between the two functions encryption and decryption. Later Base 64 decoder is used to decrypt the encrypted text and is returned in the form of a string. This decrypted string is further added to the text areas of the host and clients in our application.

This is the base of the application now we will see the GUI part of the application developed.

GUI

The GUI used in the application helps the user to customize the application according to their use providing flexibility along with safety. Use of GUI from beginning to the end-

At the start of the application the user will be asked to enter his mail id in to the Java option pane of input dialogue box.

Then in the next step the user will be asked to enter the passcode received through the mail in to the Java option pane of input dialogue box.

On success the user will be directed to a frame in white back ground with two Labels, two Text fields and a button. The labels are set with text “name”, “Chatroom name” in the case of the host and “name”, “host IP address” in the case of client. The text fields are aligned along the same line of the labels to enable the input of values through the keyboard. The button named “OK” is kept at the bottom which when clicked checks the credentials and moves on to the original application.

The application of either modes irrespective of the user consists of text field and text area. The text field is designed in the purpose of the user to type the message he wants to send in the chatroom. The text area is designed with the scroll pane which allows the users of the application to see the sent messages along with the received messages from them and all other users of the chatroom.

The send button of specific colour is aligned in middle with respect to the text field which when pressed copies the data typed in text field into the string for sending.

The uniqueness between server is identified at the top right corner of the application where the host consists of his IP address as the label visible text but all the clients consists of the chatroom name received from host on logging in into the chatroom as the visible form of label text.

The Application also consists of the menu bar which contains settings menu and font menu. The settings menu consists of the frame colour option, menu bar colour option along with the close the application option. On choosing the frame colour you will be shown a list of colours available which when clicked changes the colour of the frame which in case of menu bar colour changes the colour of menu bar totally 15x15 different combinations of unique application visibility is ensured. The Colour options available are Absolute zero, Acid Green, Aero Blue, African violet, Airforce Blue, Alabama Crimson, Alice Blue, Alloy Orange, Almond, Amarnath Deep Purple, Amarnath Pink, Amarnath Red, Amazon, Amber and Amber(SAE/ECE). The other one consists of close the application which on selecting closes the current application of the particular user.

The other menu is Font menu which consists of font colour, font style and font size options. The font colour again consists of different colour options which when selected changes the colour of the font in the text area. The colours available are same as above. The font style option consists of different font styles to be selected to change the text area font into that particular style, some of them are Times new roman, Arial, Calibri and so on. The next option that is font size option which when selected opens up a option pane asking you enter the size of the text you want, on entry the size of the text in the text area changes to that size.

In addition to these User will be shown his name on the application frame with welcome in front of his name and ‘!’ at the end of his name.

In addition the user will see one Red button with text clear chat on it along with four other orange buttons profile, participants, copy text, visit count. On selecting the button clear text the user will be asked whether is it okay to clear the text area if you press yes the text area becomes clear. On pressing the Visit count button the frame opens showing the no. of times the application has been visited so far. On pressing the copy text button it copies the entire data in the text area in to a file which later can be used in accordance with ease. On pressing the participant's button the table opens showing all the participants of the chatroom separating the user from the rest by a blank row. It shows the all participants name, host IP address and the Port through which they are connected. It also highlights the user by indicating him as the owner separating from the rest. It will be clearly seen in the screen shots part of this Report.

MERITS

- 1) The application provides Authentication support providing the security to the application.
- 2) The application uses UDP which can also be used in the cases of low latency and low bandwidth.
- 3) The Encryption-decryption techniques adopted makes the data transfer over the network safe and maintains data integrity.
- 4) The application is simple making it user friendly and is enhanced by the GUI with uniqueness for each use.
- 5) Usage of SMTP makes the application reliable and also easy to implement. The application can easily be ported from one system to other system because of the usage of the java in the application.(flexibility)

DEMERITS

- 1) Application uses UDP which does not guarantee the delivery of all the packets sent through the network, this adds as a disadvantage of using the UDP in the application.
- 2) Since UDP has no flow control, the packet may get lost or delivered twice resulting in the vulnerability of the whole application.
- 3) The usage of AES algorithm although provides data integrity, it uses simple algebraic structure that encrypts all the packets in the same way making it less efficient over the others.
- 4) The usage of SMTP makes the users mail account insecure. The application can be easily hacked and mis-used.
- 5) The usage of large amounts of GUI makes the application slow to run and load.

SOURCE CODE

The main program named PRO4 here in this implementation need some other user defined or created packages, those will be placed at the end of the source code

ORIGINAL PROGRAM

Implementing both host and client

```
import mypack.AESProgram;

import visitcount.testing;

import mailcode.mailsend;

import copytext.textcopier;

import java.awt.event.*;

import java.awt.*;

import java.net.*;

import javax.swing.*;

import java.util.*;

import java.io.*;

import java.security .SecureRandom;

import javax.swing.filechooser.FileNameExtensionFilter;

public class PRO4

{

String str1;

JLabel countsetlabel;

int visitcountinteger;

Color c1,c2,c3,c4,c5,c6,c7,c8,c9;

Color mc1,mc2,mc3,mc4,mc5,mc6,mc7,mc8,mc9,mc10,mc11,mc12,mc13,mc14,mc15;

Font mf1,mf2,mf3,mf4,mf5,mf6,mf7,mf8,mf9,mf10,mf11,mf12,mf13,mf14,mf15;

Font font1,font2,font3,font4,font5,font6;

JMenuBar mb;

JLabel l1,l2,l3,l4,l5;
```

```

JButton b1,b2,b3,b4,b5,b6;

JMenu menu,fostyle;

JMenu fm1,fm2,fm3;

JMenu i1, i2, i3;

JMenuItem cc1,cc2,cc3,cc4,cc5,cc6,cc7,cc8,cc9,cc10,cc11,cc12,cc13,cc14,cc15,cc16;

JMenuItem forclose;

JMenuItem rc1,rc2,rc3,rc4,rc5,rc6,rc7,rc8,rc9,rc10,rc11,rc12,rc13,rc14,rc15,rc16;

JMenuItem fs1,fs2,fs3,fs4,fs5,fs6,fs7,fs8,fs9,fs10,fs11,fs12,fs13,fs14,fs15,fs16;

JMenuItem zs1,zs2,zs3,zs4,zs5,zs6,zs7,zs8,zs9,zs10,zs11,zs12,zs13,zs14,zs15,zs16;

JMenuItem selsi;

JFrame f;

AESProgram aesp;

testing ting;

textcopier textcopierins;

final String XtremesecretKey = "Thisisasecretkeyextrmesecret";

public static final int HOST_MODE=0;

    public static final int CLIENT_MODE=1;

    public static final String AES = "AES";

    JButton btn_send;

    JScrollPane jScrollPane1;

    JTextArea jTextArea1;

    JLabel lbl_ipNroomName,imal;

    JTextField txt_mymsg;

    int mode;

    String Name,uhostip,chostop;

    String roomname;

    InetAddress hostip;

    DatagramSocket socket;

    ArrayList<client> ClientList;

```

```

byte[] b;

ImageIcon profileicon;

public PRO4(String myname,int mod,String ip,String room)
{try{
profileicon = new ImageIcon("commonuserphoto.jpg");
imal=new JLabel(profileicon);
aesp=new AESProgram();
ting=new testing();
textcopierins=new textcopier();
visitcountinteger=ting.countaddvisit();
f=new JFrame();

    f.setVisible(true);

    Name=myname;

    mode=mod;

    hostip=InetAddress.getByName(ip);

    chostip=ip;

    uhostip=InetAddress.getLocalHost().getHostAddress();

    roomname=room;

    f.setLayout(null);

    f.setSize(1800,1080);

    lbl_ipNroomName = new JLabel("",SwingConstants.CENTER);

    txt_mymsg = new JTextField();

    btn_send = new JButton("Send");

    jScrollPane1 = new JScrollPane();

    jTextArea1 = new JTextArea(8,15);

    ClientList=new ArrayList<>();

    f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

    f.add(lbl_ipNroomName);

```

```

lbl_ipNroomName.setBounds(1600,10,200,50);

f.add(txt_mymsg);

txt_mymsg.setBounds(50,400,600,60);

f.add(btn_send);

btn_send.setBounds(675,410,120,40);

jScrollPane1.setViewportView(jTextArea1);

f.add(jScrollPane1);

jScrollPane1.setBounds(50,475,600,450);

btn_send.setEnabled(false);

jTextArea1.setEditable(false);

txt_mymsg.setEnabled(false);

btn_send.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        String si=Name+": "+txt_mymsg.getText();

        String s=aesp.encrypt(si,XstremesecretKey);

        if(s.equals(""))==false)

        {

            if(mode==HOST_MODE)

                broadcast(s);

            else

                sendToHost(s);

            txt_mymsg.setText("");

        }

    }

});

if(mode==HOST_MODE)

{

    socket=new DatagramSocket(37988);

```



```

        lbl_ipNroomName.setText("My
IP:"+InetAddress.getLocalHost().getHostAddress());
    }
    else
    {
        socket=new DatagramSocket();
        String reqresp="!!^^"+Name+"^^!!";
        DatagramPacket pk=new
DatagramPacket(reqresp.getBytes(),reqresp.length(),hostip,37988);
        socket.send(pk);
        b=new byte[1000];
        pk=new DatagramPacket(b,1000);
        socket.setSoTimeout(6000);
        socket.receive(pk);
        reqresp=new String(pk.getData());
        if(reqresp.contains("!!^^"))
        {
            roomname=reqresp.substring(4,reqresp.indexOf("^^!!"));
            lbl_ipNroomName.setText("ChatRoom: "+roomname);
            btn_send.setEnabled(true);
            txt_mymsg.setEnabled(true);
        }
        else{
            JOptionPane.showMessageDialog(f,"No response from the
server");System.exit(0);
        }
    }
    Messenger.start();

```

```
c1=new Color(255,126,0);  
c2=new Color(124,185,232);  
c3=new Color(240,248,255);  
c4=new Color(114,160,193);  
c5=new Color(75,110,152);  
c6=new Color(0,0,0);
```

```
mc1=new Color(0,72,186);  
mc2=new Color(176,191,26);  
mc3=new Color(201,255,229);  
mc4=new Color(178,132,190);  
mc5=new Color(93,138,168);  
mc6=new Color(175,0,42);  
mc7=new Color(240,248,255);  
mc8=new Color(196,98,16);  
mc9=new Color(239,222,205);  
mc10=new Color(171,39,79);  
mc11=new Color(241,156,187);  
mc12=new Color(211,33,45);  
mc13=new Color(59,122,87);  
mc14=new Color(255,191,0);  
mc15=new Color(255,126,0);
```

```
mb=new JMenuBar();  
font1 = new Font("Verdana", Font.BOLD, 18);  
font2 = new Font("Courier", Font.PLAIN, 16);  
font3 = new Font("Castellar", Font.BOLD, 24);  
font4 = new Font("Yellowtail", Font.BOLD, 88);
```

```

font5 = new Font("Dungeon", Font.BOLD, 92);

b1=new JButton();
b2=new JButton();
b3=new JButton();
b4=new JButton();
b5=new JButton();

//label 1
l1=new JLabel("Welcome");
l1.setFont(font4);
l1.setBounds(300,200,500,100);

//label 2
l2=new JLabel(Name+"!");
l2.setFont(font5);
l2.setBounds(700,200,900,100);
l2.setForeground(c3);


//button 1
b1.setBounds(50,50,300,80);
b1.setBackground(mc12);
b1.setText("CLEAR CHAT");
b1.setFont(font3);

//button2
b2.setBounds(400,50,300,80);
b2.setBackground(c1);
b2.setText("PROFILE");
b2.setFont(font3);

//button3
b3.setBounds(750,50,300,80);
b3.setBackground(c1);

```

```

b3.setText("PARTICIPANTS");
b3.setFont(font3);
//button4
b4.setBounds(1100,50,300,80);
b4.setBackground(c1);
b4.setText("COPY TEXT TO FILE");
b4.setFont(font3);
//button 5
b5.setBounds(1450,50,300,80);
b5.setBackground(c1);
b5.setText("VISIT COUNT");
b5.setFont(font3);

//button controls
//button1
b1.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{

        String temp1="Are you sure,You want to clear the Chat Window ?"+" \n "+Name;
        int oi1=JOptionPane.showConfirmDialog(f,temp1);
        if(oi1==JOptionPane.YES_OPTION){
            jTextArea1.setText(" ");
        }

    }
    catch(Exception w){}
    }
});

```

```
//button2  
  
b2.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        profile();  
    }  
    catch(Exception w){}  
    }  
});
```

```
//button 3  
  
b3.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
  
        participants();  
  
    }  
    catch(Exception w){}  
    }  
});
```

```
//button4  
  
b4.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)
```

```

    { try{
gametime();

    }
    catch(Exception w){}
    }
});

//button 5

b5.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        viewcount();
    }
    catch(Exception w){}
    }
});

//menu controls

menu=new JMenu("Settings");
fostyle=new JMenu("Font");
menu.setFont(font1);
fostyle.setFont(font1);
mb.setBackground(c5);
menu.setPreferredSize(new Dimension(100, 100));
fostyle.setPreferredSize(new Dimension(100, 100));

```

```
i1=new JMenu("Frame Color");  
i1.setFont(font2);  
i2=new JMenu("Menu bar color");  
i2.setFont(font2);  
i3=new JMenu("Close");  
i3.setFont(font2);
```

```
fm1=new JMenu("Font Style");  
fm2=new JMenu("Font Size");  
fm3=new JMenu("Font Color");  
fm1.setFont(font2);  
fm2.setFont(font2);  
fm3.setFont(font2);
```

```
mb.add(menu);  
menu.add(i1);  
menu.add(i2);  
menu.add(i3);  
fostyle.add(fm1);  
fostyle.add(fm2);  
fostyle.add(fm3);  
mb.add(fostyle);  
f.setJMenuBar(mb);
```

```
//close part  
forclose=new JMenuItem("close the application");  
i3.add(forclose);  
forclose.setFont(font2);
```

```
//menuitem controls
```

```
cc1=new JMenuItem("Absolute zero");  
cc2=new JMenuItem("Acid Green");  
cc3=new JMenuItem("Aero Blue");  
cc4=new JMenuItem("African Violet");  
cc5=new JMenuItem("Airforce Blue");  
cc6=new JMenuItem("Alabama Crimson");  
cc7=new JMenuItem("Alice Blue");  
cc8=new JMenuItem("Alloy Orange");  
cc9=new JMenuItem("Almond");  
cc10=new JMenuItem("Amarnath Deep Purple");  
cc11=new JMenuItem("Amarnath Pink");  
cc12=new JMenuItem("Amarnath Red");  
cc13=new JMenuItem("Amazon");  
cc14=new JMenuItem("Amber");  
cc15=new JMenuItem("Amber(SAE/ECE)");  
cc16=new JMenuItem("Default Color");
```

```
rc1=new JMenuItem("Absolute zero");  
rc2=new JMenuItem("Acid Green");  
rc3=new JMenuItem("Aero Blue");  
rc4=new JMenuItem("African Violet");  
rc5=new JMenuItem("Airforce Blue");  
rc6=new JMenuItem("Alabama Crimson");  
rc7=new JMenuItem("Alice Blue");  
rc8=new JMenuItem("Alloy Orange");  
rc9=new JMenuItem("Almond");  
rc10=new JMenuItem("Amarnath Deep Purple");
```



```
rc11=new JMenuItem("Amarnath Pink");
rc12=new JMenuItem("Amarnath Red");
rc13=new JMenuItem("Amazon");
rc14=new JMenuItem("Amber");
rc15=new JMenuItem("Amber(SAE/ECE)");
rc16=new JMenuItem("Default Color");
```

```
i1.add(cc16);
i1.add(cc1);
i1.add(cc2);
i1.add(cc3);
i1.add(cc4);
i1.add(cc5);
i1.add(cc6);
i1.add(cc7);
i1.add(cc8);
i1.add(cc9);
i1.add(cc10);
i1.add(cc11);
i1.add(cc12);
i1.add(cc13);
i1.add(cc14);
i1.add(cc15);
```

```
i2.add(rc16);
i2.add(rc1);
i2.add(rc2);
i2.add(rc3);
i2.add(rc4);
```

```
i2.add(rc5);  
i2.add(rc6);  
i2.add(rc7);  
i2.add(rc8);  
i2.add(rc9);  
i2.add(rc10);  
i2.add(rc11);  
i2.add(rc12);  
i2.add(rc13);  
i2.add(rc14);  
i2.add(rc15);
```

```
cc16.setFont(font2);  
cc1.setFont(font2);  
cc2.setFont(font2);  
cc3.setFont(font2);  
cc4.setFont(font2);  
cc5.setFont(font2);  
cc6.setFont(font2);  
cc7.setFont(font2);  
cc8.setFont(font2);  
cc9.setFont(font2);  
cc10.setFont(font2);  
cc11.setFont(font2);  
cc12.setFont(font2);  
cc13.setFont(font2);  
cc14.setFont(font2);  
cc15.setFont(font2);
```

```
rc16.setFont(font2);  
rc1.setFont(font2);  
rc2.setFont(font2);  
rc3.setFont(font2);  
rc4.setFont(font2);  
rc5.setFont(font2);  
rc6.setFont(font2);  
rc7.setFont(font2);  
rc8.setFont(font2);  
rc9.setFont(font2);  
rc10.setFont(font2);  
rc11.setFont(font2);  
rc12.setFont(font2);  
rc13.setFont(font2);  
rc14.setFont(font2);  
rc15.setFont(font2);
```

```
//fostyle menuitems part
```

```
mf1=new Font("Arial", Font.PLAIN, 16);  
mf2=new Font("Calibri", Font.PLAIN, 16);  
mf3=new Font("Candara", Font.PLAIN, 16);  
mf4=new Font("Cambria", Font.PLAIN, 16);  
mf5=new Font("Dungeon", Font.PLAIN, 16);  
mf6=new Font("Ebrima", Font.PLAIN, 16);  
mf7=new Font("Gabriola", Font.PLAIN, 16);  
mf8=new Font("Fixedsys", Font.PLAIN, 16);  
mf9=new Font("Forte", Font.PLAIN, 16);  
mf10=new Font("Harrington", Font.PLAIN, 16);
```

```
mf11=new Font("Jokerman", Font.PLAIN, 16);  
mf12=new Font("Impact", Font.PLAIN, 16);  
mf13=new Font("Leelawadee", Font.PLAIN, 16);  
mf14=new Font("Kalam", Font.PLAIN, 16);  
mf15=new Font("Magneto", Font.PLAIN, 16);
```

```
fs1=new JMenuItem("Arial");  
fs2=new JMenuItem("Calibri");  
fs3=new JMenuItem("Candara");  
fs4=new JMenuItem("Cambria");  
fs5=new JMenuItem("Dungeon");  
fs6=new JMenuItem("Ebrima");  
fs7=new JMenuItem("Gabriola");  
fs8=new JMenuItem("Fixedsys");  
fs9=new JMenuItem("Forte");  
fs10=new JMenuItem("Harrington");  
fs11=new JMenuItem("Jokerman");  
fs12=new JMenuItem("Impact");  
fs13=new JMenuItem("Leelawadee");  
fs14=new JMenuItem("Kalam");  
fs15=new JMenuItem("Magneto");  
fs16=new JMenuItem("Default settings");  
  
//adding  
fm1.add(fs16);  
fm1.add(fs1);  
fm1.add(fs2);  
fm1.add(fs3);  
fm1.add(fs4);  
fm1.add(fs5);
```

```
fm1.add(fs6);  
fm1.add(fs7);  
fm1.add(fs8);  
fm1.add(fs9);  
fm1.add(fs10);  
fm1.add(fs11);  
fm1.add(fs12);  
fm1.add(fs13);  
fm1.add(fs14);  
fm1.add(fs15);
```

```
fs16.setFont(font2);  
fs1.setFont(mf1);  
fs2.setFont(mf2);  
fs3.setFont(mf3);  
fs4.setFont(mf4);  
fs5.setFont(mf5);  
fs6.setFont(mf6);  
fs7.setFont(mf7);  
fs8.setFont(mf8);  
fs9.setFont(mf9);  
fs10.setFont(mf10);  
fs11.setFont(mf11);  
fs12.setFont(mf12);  
fs13.setFont(mf13);  
fs14.setFont(mf14);  
fs15.setFont(mf15);
```

```
//font color part
```

```

zs1=new JMenuItem("Absolute zero");
zs2=new JMenuItem("Acid Green");
zs3=new JMenuItem("Aero Blue");
zs4=new JMenuItem("African Violet");
zs5=new JMenuItem("Airforce Blue");
zs6=new JMenuItem("Alabama Crimson");
zs7=new JMenuItem("Alice Blue");
zs8=new JMenuItem("Alloy Orange");
zs9=new JMenuItem("Almond");
zs10=new JMenuItem("Amarnath Deep Purple");
zs11=new JMenuItem("Amarnath Pink");
zs12=new JMenuItem("Amarnath Red");
zs13=new JMenuItem("Amazon");
zs14=new JMenuItem("Amber");
zs15=new JMenuItem("Amber(SAE/ECE)");
zs16=new JMenuItem("Default Color");

//adding
fm3.add(zs16);
fm3.add(zs1);
fm3.add(zs2);
fm3.add(zs3);
fm3.add(zs4);
fm3.add(zs5);
fm3.add(zs6);
fm3.add(zs7);
fm3.add(zs8);
fm3.add(zs9);
fm3.add(zs10);

```

```
fm3.add(zs11);
```

```
fm3.add(zs12);
```

```
fm3.add(zs13);
```

```
fm3.add(zs14);
```

```
fm3.add(zs15);
```

```
zs16.setFont(font2);
```

```
zs1.setFont(font2);
```

```
zs2.setFont(font2);
```

```
zs3.setFont(font2);
```

```
zs4.setFont(font2);
```

```
zs5.setFont(font2);
```

```
zs6.setFont(font2);
```

```
zs7.setFont(font2);
```

```
zs8.setFont(font2);
```

```
zs9.setFont(font2);
```

```
zs10.setFont(font2);
```

```
zs11.setFont(font2);
```

```
zs12.setFont(font2);
```

```
zs13.setFont(font2);
```

```
zs14.setFont(font2);
```

```
zs15.setFont(font2);
```

```
//size
```

```
selsi=new JMenuItem("Select");
```

```
fm2.add(selsi);
```

```
selsi.setFont(font2);
```

```

//Menu Colors part

cc1.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc1);
    }
    catch(Exception w){}
    }
});

cc2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc2);
    }
    catch(Exception w){}
    }
});

cc3.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc3);
    }
    catch(Exception w){}
    }
});

```



```
cc4.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        f.getContentPane().setBackground(mc4);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
cc5.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        f.getContentPane().setBackground(mc5);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
cc6.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        f.getContentPane().setBackground(mc6);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
cc7.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)
```

```

{ try{
    f.getContentPane().setBackground(mc7);
}
catch(Exception w){}
}
});

```

```

cc8.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc8);
    }
    catch(Exception w){}
    }
});

```

```

cc9.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc9);
    }
    catch(Exception w){}
    }
});

```

```

cc10.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc10);

```

```

    }
    catch(Exception w){}
    }
});

cc11.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc11);
    }
    catch(Exception w){}
    }
});

cc12.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc12);
    }
    catch(Exception w){}
    }
});

cc13.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc13);
    }
    catch(Exception w){}

```

```

    }
});

cc14.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc14);
    }
    catch(Exception w){}
    }
});

cc15.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(mc15);
    }
    catch(Exception w){}
    }
});

cc16.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        f.getContentPane().setBackground(c4);
    }
    catch(Exception w){}
    }
});

```

```

//menubar color listeners

rc1.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc1);
    }
    catch(Exception w){ }
    }
});

rc2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc2);
    }
    catch(Exception w){ }
    }
});

rc3.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc3);
    }
    catch(Exception w){ }
    }
});

```

```
rc4.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        mb.setBackground(mc4);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
rc5.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        mb.setBackground(mc5);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
rc6.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        mb.setBackground(mc6);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
rc7.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)
```

```
{ try{  
    mb.setBackground(mc7);  
}  
catch(Exception w){}  
}  
});
```

```
rc8.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        mb.setBackground(mc8);  
    }  
    catch(Exception w){}  
    }  
});
```

```
rc9.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        mb.setBackground(mc9);  
    }  
    catch(Exception w){}  
    }  
});
```

```
rc10.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        mb.setBackground(mc10);
```

```

    }
    catch(Exception w){}
    }
});

rc11.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc11);
    }
    catch(Exception w){}
    }
});

rc12.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc12);
    }
    catch(Exception w){}
    }
});

rc13.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc13);
    }
    catch(Exception w){}

```



```

    }
});

rc14.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc14);
    }
    catch(Exception w){}
    }
});

```

```

rc15.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(mc15);
    }
    catch(Exception w){}
    }
});

```

```

rc16.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        mb.setBackground(c5);
    }
    catch(Exception w){}
    }
});

```

//Font style Action Listener part

```
fs16.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        jTextArea1.setFont(font2);  
    }  
    catch(Exception w){}  
    }  
});
```

```
fs1.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        jTextArea1.setFont(mf1);  
    }  
    catch(Exception w){}  
    }  
});
```

```
fs1.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        jTextArea1.setFont(mf1);  
    }  
    catch(Exception w){}  
    }  
});
```

```
fs2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        JTextArea1.setFont(mf1);
    }
    catch(Exception w){ }
    }
});
```

```
fs1.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        JTextArea1.setFont(mf2);
    }
    catch(Exception w){ }
    }
});
```

```
fs3.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        JTextArea1.setFont(mf3);
    }
    catch(Exception w){ }
    }
});
```

```
fs4.addActionListener(new ActionListener(){
```

```

    public void actionPerformed(ActionEvent ae)
    { try{
        JTextArea1.setFont(mf4);
    }
    catch(Exception w){}
    }
});

```

```

fs5.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        JTextArea1.setFont(mf5);
    }
    catch(Exception w){}
    }
});

```

```

fs6.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        JTextArea1.setFont(mf6);
    }
    catch(Exception w){}
    }
});

```

```

fs7.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{

```

```

        jTextArea1.setFont(mf7);
    }
    catch(Exception w){ }
}
});

fs8.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setFont(mf8);
    }
    catch(Exception w){ }
    }
});

fs9.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setFont(mf9);
    }
    catch(Exception w){ }
    }
});

fs10.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setFont(mf10);
    }

```

```

        catch(Exception w){}

    }

});

fs11.addActionListener(new ActionListener(){

    public void actionPerformed(ActionEvent ae)

    { try{

        JTextArea1.setFont(mf11);

    }

    catch(Exception w){}

    }

});

```

```

fs12.addActionListener(new ActionListener(){

    public void actionPerformed(ActionEvent ae)

    { try{

        JTextArea1.setFont(mf12);

    }

    catch(Exception w){}

    }

});

```

```

fs13.addActionListener(new ActionListener(){

    public void actionPerformed(ActionEvent ae)

    { try{

        JTextArea1.setFont(mf13);

    }

    catch(Exception w){}

    }

});

```

```
});
```

```
fs14.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        jTextArea1.setFont(mf14);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
fs15.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        jTextArea1.setFont(mf15);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
//Color of the Font
```

```
zs16.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        jTextArea1.setForeground(c6);  
    }  
    catch(Exception w){ }  
    }  
}
```

```

});

zs1.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc1);
    }
    catch(Exception w){ }
    }
});

```

```

zs2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc2);
    }
    catch(Exception w){ }
    }
});

```

```

zs3.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc3);
    }
    catch(Exception w){ }
    }
});

```



```
zs4.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        JTextArea1.setForeground(mc4);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
zs5.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        JTextArea1.setForeground(mc5);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
zs6.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)  
    { try{  
        JTextArea1.setForeground(mc6);  
    }  
    catch(Exception w){ }  
    }  
});
```

```
zs7.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent ae)
```

```

{ try{
    jTextArea1.setForeground(mc7);
}
catch(Exception w){}
}
});

zs8.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc8);
    }
    catch(Exception w){}
    }
});

zs9.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc9);
    }
    catch(Exception w){}
    }
});

zs10.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc10);
    }
    catch(Exception w){}
    }
});

```

```

    }
    catch(Exception w){}
    }
});

zs11.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc11);
    }
    catch(Exception w){}
    }
});

zs12.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc12);
    }
    catch(Exception w){}
    }
});

zs13.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc13);
    }
    catch(Exception w){}

```

```

    }
});

zs14.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc14);
    }
    catch(Exception w){}
    }
});

```

```

zs15.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        jTextArea1.setForeground(mc15);
    }
    catch(Exception w){}
    }
});

```

```

//setting font size

```

```

selsi.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        String tem2=JOptionPane.showInputDialog(f,"Enter the Size b/w 2 to 100 only in
integers"+" "+Name);
        int rou2=Integer.parseInt(tem2);
    }
    catch(Exception w){}
    }
});

```

```

        Font tempstoref = jTextArea1.getFont();
        Font tempstoref2=new Font(tempstoref.getFontName(), tempstoref.getStyle(),rou2);
        jTextArea1.setFont(tempstoref2);
    }
    catch(Exception w){ }
}
});

//closing part in menu
forclose.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{
        int askingforconfirm=JOptionPane.showConfirmDialog(f,"Are you sure? you want
to close the application "+Name);
        if(askingforconfirm==JOptionPane.YES_OPTION){
            System.exit(0);
        }
    }
    catch(Exception w){ }
}
});

//common
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setLayout(null);
f.add(b1);
f.add(b2);
f.add(b3);
f.add(b4);

```

```

f.add(b5);

f.add(l1);

f.add(l2);

f.getContentPane().setBackground(c4);

f.setVisible(true);

} catch (Exception ex) { JOptionPane.showMessageDialog(null, ex); }

}

public static void main(String args[])
{
    mailsend security=new mailsend();

    String mailidinput=JOptionPane.showInputDialog("Enter Your gmail id for login in
to the application");

    if ((mailidinput.contains("@gmail.com"))==false)
    {JOptionPane.showInputDialog("User input invalid.....only gmail id accpeted");}

    else{

        String trup=security.sendmail(mailidinput);

        String micodev=JOptionPane.showInputDialog("Message sent successfully \n
Please enter the passcode received via mail to proceed further in to the application");

        if(!trup.equals(micodev))
        {

            JOptionPane.showInputDialog("Enter the Valid code to proceed or else if you didn't
receive the code try closing and mentioning the correct email address while logging
in.....");

        }

        else{

            try {

                int mode=JOptionPane.showConfirmDialog(null,"Create a chatroom or connect to
existing one?\nYes - Create Chat Room\nNo - Jion a Chat Room","Create or
Join?",JOptionPane.YES_NO_OPTION);

                JFrame datainp=new JFrame("INFO INPUT");

```

```
Font mafa1=new Font("Verdana", Font.BOLD, 18);
```

```
Font mafa2=new Font("Calibri", Font.BOLD, 20);
```

```
Color infoinputcolor=new Color(255,255,255);
```

```
JLabel nala=new JLabel("Name");
```

```
JLabel iala=new JLabel("Hostip address");
```

```
JLabel cnla=new JLabel("chatroom name");
```

```
nala.setFont(mafa1);
```

```
iala.setFont(mafa1);
```

```
cnla.setFont(mafa1);
```

```
JTextField nafi=new JTextField();
```

```
JTextField iaifi=new JTextField();
```

```
JTextField cnfi=new JTextField();
```

```
nafi.setFont(mafa2);
```

```
iaifi.setFont(mafa2);
```

```
cnfi.setFont(mafa2);
```

```
JButton acbu=new JButton("ok");
```

```
acbu.setFont(mafa2);
```

```
nala.setBounds(70,200,200,100);
```

```
datainp.add(nala);
```

```
nafi.setBounds(300,220,400,50);
```

```
datainp.add(nafi);
```

```
acbu.setBounds(340,600,100,60);
```

```

datainp.add(acbu);
if(mode==1)
{
iala.setBounds(70,350,200,100);
datainp.add(iala);
iafi.setBounds(300,370,400,50);
datainp.add(iafi);

}
else
{
cnla.setBounds(70,350,200,100);
datainp.add(cnla);
cnfi.setBounds(300,370,400,50);
datainp.add(cnfi);
}

acbu.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    {
        String host="",room="";
        String name=nafi.getText();
        if(name==null||name.equals(""))
            {JOptionPane.showMessageDialog(datainp, "Name cannot be blank");return;}
        try
        {
            if(mode==1)
            {
                host=iafi.getText();

```



```

        if(host==null||host.equals(""))
            {JOptionPane.showMessageDialog(datainp, "IP of host is mandatory");return;}
        }
    else
    {
        room=cnfi.getText();

    }

    PRO4 obj= new PRO4(name,mode,host,room);
    }
    catch(Exception w){ }
    }
});

datainp.getContentPane().setBackground(infoinputcolor);
datainp.setSize(800,800);
datainp.setLayout(null);
datainp.setVisible(true);

    } catch (Exception ex) {JOptionPane.showMessageDialog(null,ex);}
}}
}

public void broadcast(String str)
{
    try {

```

```

DatagramPacket pack=new DatagramPacket(str.getBytes(),str.length());
for(int i=0;i<ClientList.size();i++)
{
    pack.setAddress(InetAddress.getByName(ClientList.get(i).ip));
    pack.setPort(ClientList.get(i).port);
    socket.send(pack);
}

//decryption of str required before adding it to the text area
String hur=str.trim();
String dcs;
dcs=aesp.decrypt(hur,XstremesecretKey) ;

jTextArea1.setText(jTextArea1.getText()+"\n"+dcs);
} catch (Exception ex) {JOptionPane.showMessageDialog(f,ex);}
}

public void sendToHost(String str)
{
    DatagramPacket pack=new DatagramPacket(str.getBytes(),str.length(),hostip,37988);
    try {socket.send(pack);} catch (Exception ex)
    {JOptionPane.showMessageDialog(f,"Sending to server failed");}
}

Thread Messenger=new Thread()
{
    public void run()
    {
        try {

```

```

while(true)
{
    b=new byte[300];
    DatagramPacket pkt=new DatagramPacket(b,300);
    socket.setSoTimeout(0);
    socket.receive(pkt);
    String s=new String(pkt.getData());
    if(mode==HOST_MODE)
    {
        if(s.contains("!!^^"))
        {
            client temp=new client();
            temp.ip=pkt.getAddress().getHostAddress();
            temp.port=pkt.getPort();
            temp.name=s.substring(4,s.indexOf("^^!!"));
            String xyz=s.substring(4,s.indexOf("^^!!"))+" joined.";

            String ter=aesp.encrypt(xyz,XstremesecretKey);

            broadcast(ter);
            ClientList.add(temp);

            s="!!^^"+roomname+"^^!!";

            pkt=new
DatagramPacket(s.getBytes(),s.length(),InetAddress.getByName(temp.ip),temp.port);
            socket.send(pkt);
            btn_send.setEnabled(true);
            txt_mymsg.setEnabled(true);

```



```

praname1=new JLabel(" Name:");
praname1.setFont(imahefont);
edli=new JLabel("Edit");
edli.setBounds(640,18,40,40);
praname2=new JLabel(Name);
praname2.setFont(imaansfont);

prochat1=new JLabel("Chat Room Name:");
proip1=new JLabel("IP Address:");

prochat2=new JLabel(roomname);
proip2=new JLabel();

if(mode==HOST_MODE)
{
proip2.setText(uhostip);
}
else
{
proip2.setText(chostip);
}
prochat1.setFont(imahefont);
proip1.setFont(imahefont);

prochat2.setFont(imaansfont);
proip2.setFont(imaansfont);

praname1.setBounds(80,400,338,60);
praname2.setBounds(420,402,360,70);

```

```

prochat1.setBounds(80,470,338,60);
prochat2.setBounds(420,472,360,70);

proip1.setBounds(80,540,338,60);
proip2.setBounds(420,542,360,70);

imal.setBounds(175,0,400,400);

imabu.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae)
    { try{

        JFileChooser profileimagechoose = new JFileChooser();
profileimagechoose.setCurrentDirectory(new File(System.getProperty("user.home")));
        FileNameExtensionFilter filter = new FileNameExtensionFilter("*.Images",
"jpg","gif","png");
        profileimagechoose.addChoosableFileFilter(filter);
        int imageresult = profileimagechoose.showSaveDialog(null);
        if(imageresult == JFileChooser.APPROVE_OPTION){
            File selectedFile = profileimagechoose.getSelectedFile();
            String path = selectedFile.getAbsolutePath();
            imal.setIcon(ResizeImage(path));}
        else if(imageresult == JFileChooser.CANCEL_OPTION){
            JOptionPane.showMessageDialog(q,"No file select");
        }
    }
    else{ }
}

```

```

    }
    catch(Exception w){}
    }
});

q.getContentPane().setBackground(whitecolor);
q.add(imal);
q.add(edli);
q.add(proname1);
q.add(proname2);
q.add(prochat1);
q.add(prochat2);
q.add(proip1);
q.add(proip2);
q.add(imabu);
q.setLayout(null);
q.setSize(800,800);
q.setVisible(true);
q.setResizable(false);
}
catch(Exception d)
{
JOptionPane.showMessageDialog(q,d);
}
}

//method for profile image selection
public ImageIcon ResizeImage(String ImagePath)

```

```

{
    ImageIcon MyImage = new ImageIcon(ImagePath);

    Image img = MyImage.getImage();

    Image newImg = img.getScaledInstance(imal.getWidth(), imal.getHeight(),
Image.SCALE_SMOOTH);

    ImageIcon image = new ImageIcon(newImg);

    return image;
}

```

```

public void participants()

```

```

{try{

```

```

    JFrame g=new JFrame();

```

```

    g.setSize(600,600);

```

```

    g.setVisible(true);

```

```

    String qtem1,qtem3,px;

```

```

    int qtem2;

```

```

    int pj=10;

```

```

    JTable jtble;

```

```

    JOptionPane.showMessageDialog(g,"You will be able to see all the people in the
chatRoom only if you are a Host.If not you can only see yourself in the list");

```

```

    String[] columnNames = { "Name", "IP adress", "Port number" };

```

```

    String[][] dataadd=new String[100][100];

```

```

    int i;

```

```

    for(i=0;i<ClientList.size();i++)

```

```

    {

```

```

        qtem1=ClientList.get(i).ip;

```

```

        qtem2=ClientList.get(i).port;

```

```

        qtem3=ClientList.get(i).name;

```



```

        dataadd[i][0]=qtem3;
        dataadd[i][1]=qtem1;
        dataadd[i][2]=String.valueOf(qtem2);

    }

    dataadd[i+1][0]=Name+"( owner )";
    dataadd[i+1][1]=uhostip;
    dataadd[i+1][2]=String.valueOf(37988);
    jTable = new JTable(dataadd, columnNames);
    jTable.setBounds(50, 40, 400, 400);
    JScrollPane spyt = new JScrollPane(jTable);
    g.add(spyt);
    g.setResizable(false);
} catch (Exception e) { }
}

public void gametime()
{
    String forco=jTextArea1.getText();
    int rejoy=textcopierins.copy_text(forco);
    if(rejoy==1)
    JOptionPane.showMessageDialog(f,"Text copied into file textcopy.txt
    SUCCESSFULLY"+Name);
    else
    JOptionPane.showMessageDialog(f,"Some mistake in copying once check "+Name);
}

public void viewcount()
{try{

```

```

Color whitecolor=new Color(255,255,255);
countsetlabel=new JLabel();
Font vicfont= new Font("Helvetica",Font.BOLD,32);
JFrame countframe=new JFrame("VIEW COUNT");
countsetlabel.setText("COUNT : "+String.valueOf(visitcountinteger));
countsetlabel.setBounds(0,20,200,100);
countsetlabel.setFont(vicfont);
countframe.add(countsetlabel);
countframe.setLayout(null);
countframe.setVisible(true);
countframe.setSize(200,200);
countframe.setResizable(false);
countframe.getContentPane().setBackground(whitecolor);
}
catch(Exception e){ }
}

}

class client
{
public String ip;
public int port;
public String name;
}

```

USER-DEFINED PACKAGES AND OTHER PROGRAMS USED IN THE MAIN PROGRAM'S SOURCE CODE

Will be of the form x.y where x is the Package and y is the objectclass

Visitcount.testing Code

```
package visitcount;

import java.io.File;
import java.io.*;
import java.util.Scanner;

public class testing
{
    public int countaddvisit()
    {int yc=0;
    try{
        File inputFile = new File("E:viewcount.txt");
        if (!inputFile.isFile()) {
            System.out.println("Parameter is not an existing file");
        }

        File tempFile = new File(inputFile.getAbsolutePath() + ".tmp");
        BufferedReader br = new BufferedReader(new FileReader("E:viewcount.txt"));
        PrintWriter pw = new PrintWriter(new FileWriter(tempFile));

        String line = null;
        line = br.readLine();
        int countadd=Integer.parseInt(line);
        int temp=++countadd;
        yc=temp;
        String newline=Integer.toString(temp);
        pw.println(newline);
        pw.flush();
```

```

        pw.close();
        br.close();

        if (!inputFile.delete()) {
            System.out.println("Could not delete file");
        }

        if (!tempFile.renameTo(inputFile))
            System.out.println("Could not rename file");

    }
    catch(Exception r){ }
    return yc;
}

public static void main(String args[])
{
    try{
        testing t=new testing();
        t.countaddvisit();
    }
    catch(Exception er){ }
}
}

```

mypack.AESProgram code

```
package mypack;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AESProgram{

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey)
    {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        }
        catch (NoSuchAlgorithmException k) {
            k.printStackTrace();
        }
        catch (UnsupportedEncodingException k) {
            k.printStackTrace();
        }
    }
}
```

```

public static String encrypt(String strToEncrypt, String secret)
{
    String encry_var=new String();
    try
    {
        setKey(secret);

        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);

        encry_var=
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
    }
    catch (Exception e)
    {
        System.out.println("Error while encrypting: " + e.toString());
    }
    return encry_var;
}

```

```

public static String decrypt(String strToDecrypt, String secret)
{
    String decry_var=new String();
    try
    {
        setKey(secret);

        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);

        decry_var=new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
    }
}

```

```

        catch (Exception e)
        {
            System.out.println("Error while decrypting: " + e.toString());
        }

        return decry_var;
    }

    public static void main(String args[])
    {
        System.out.println("Welcome to the encryption and decryption AESProgram class of
        mypack Package");
    }
}

```

mailcode.mailsend code

```

package mailcode;

import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import javax.activation.*;
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.event.*;

import generatornumber.randomnumbergenerator;


public class mailsend
{
    String universalrandomstore;

    String to;

```

```
randomnumbergenerator rn6g ;
```

```
public String sendmail(String dx){  
    rn6g=new randomnumbergenerator();  
    universalrandomstore=rn6g.passrandom();  
    String to = dx;  
    String from = "madireddykalyanvenkat@gmail.com";  
    Properties properties = new Properties();  
    properties.put("mail.smtp.auth", "true");  
    properties.put("mail.smtp.starttls.enable", "true");  
    properties.put("mail.smtp.host", "smtp.gmail.com");  
    properties.put("mail.smtp.port", 587);  
    Session session = Session.getDefaultInstance(properties,new  
    javax.mail.Authenticator()  
    {  
        protected PasswordAuthentication getPasswordAuthentication(){  
            return new  
PasswordAuthentication("madireddykalyanvenkat@gmail.com","xxxxxxx");  
        }  
    });  
    try{  
        MimeMessage message = new MimeMessage(session);  
        message.setFrom(new InternetAddress(from));  
        message.addRecipient(Message.RecipientType.TO,new InternetAddress(to));  
        message.setSubject("Log-In Confirmation");  
        message.setText(universalrandomstore);  
        Transport.send(message);  
        System.out.println("Message sent successfully");  
    }catch (MessagingException mex) {mex.printStackTrace();}  
    return universalrandomstore;}}
```



Pass-
word of
the mail

copytext.textcopier code

```
package copytext;

import java.io.FileWriter;

public class textcopier {

    public int copy_text(String xs)
    {
        try{
            FileWriter fw=new FileWriter("E:\\textcopy.txt");
            fw.write(xs);
            fw.close();
        }catch(Exception e){System.out.println(e);}

        return 1;
    }
}
```

generatordnumber.randomnumbergenerator code

This is used by the mail send code which is also a package.

```
package generatordnumber;

public class randomnumbergenerator {

    int n;

    public randomnumbergenerator()
    {
        n=6;
    }

    static String getAlphanumericString(int n)
    {
```

```

        String AlphaNumericString =
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ" + "0123456789" +
        "abcdefghijklmnopqrstuvwxyz";

```

```

        StringBuilder sb = new StringBuilder(n);

```

```

        for (int i = 0; i < n; i++) {

```

```

            int index

```

```

                = (int)(AlphaNumericString.length()

```

```

                    * Math.random());

```

```

                sb.append(AlphaNumericString

```

```

                    .charAt(index));

```

```

            }

```

```

        return sb.toString();

```

```

    }

```

```

public String passrandom()

```

```

{

```

```

    String passtemp=randomnumbergenerator.getAlphaNumericString(n);

```

```

    return passtemp;

```

```

}

```

SNAPSHOT'S

First run the application as many times to create as many users

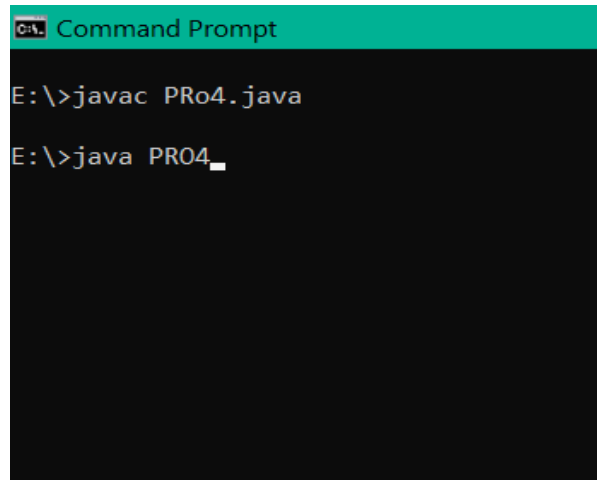


Fig. 3.1.application run.

Then you will be asked to enter the mail-address like this

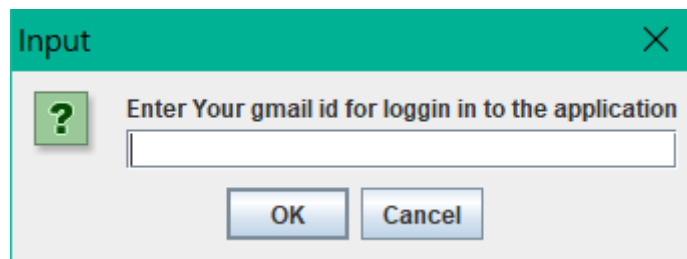


Fig. 3.2.mail id pane.

Enter your mail-address and click ok

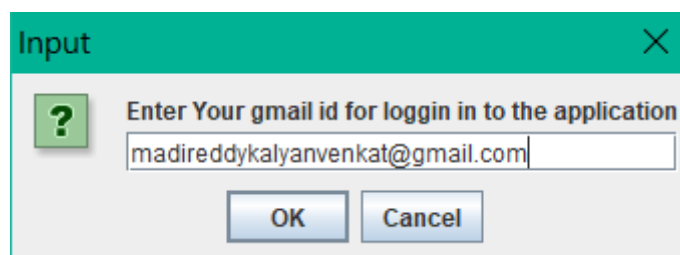


Fig. 3.3.mail id entry pane.

On clicking okay you will be asked to enter the pass code like this

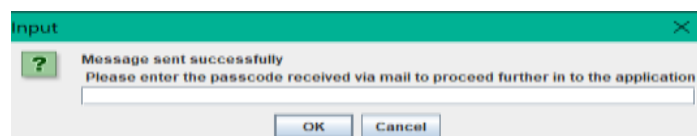


Fig. 3.4.passcode pane.

Check the mail address specified to check the code received

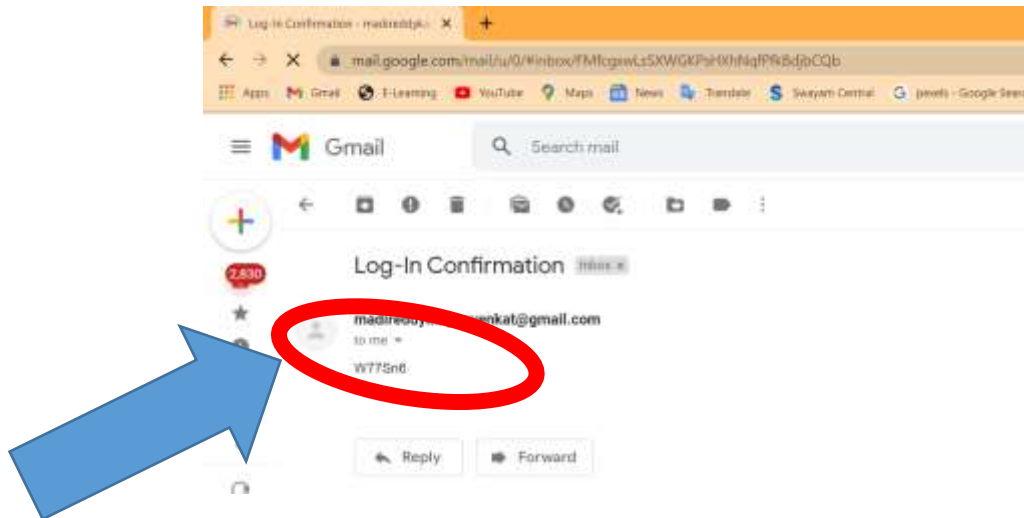


Fig. 3.5.mail screen shot.

After that enter that code in the asked option pane and click ok

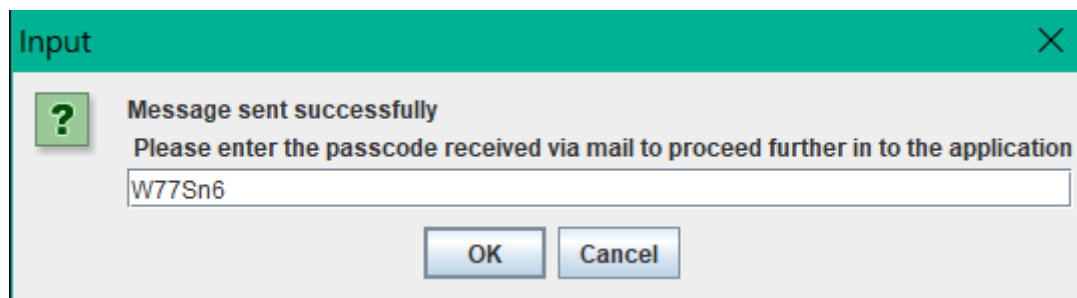


Fig. 3.6.pass code entry.

On entering you will be asked whether to behave as a host or client

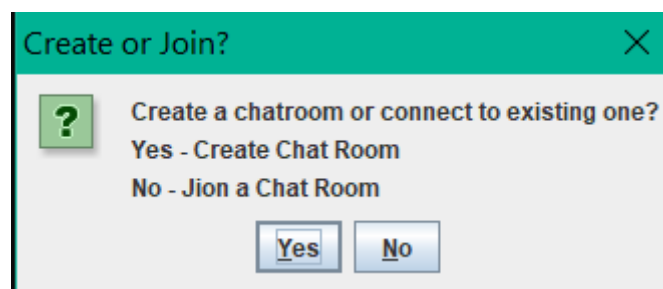
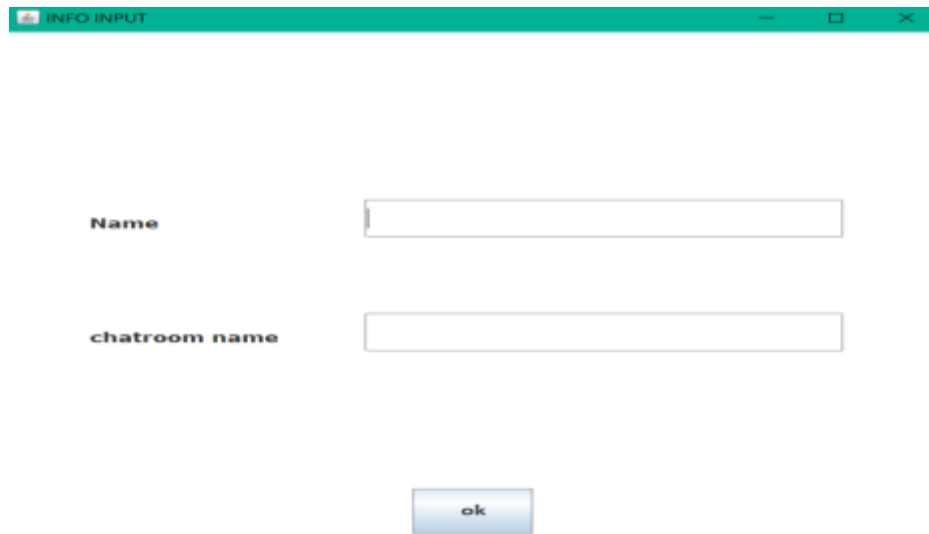


Fig. 3.7.host or client.

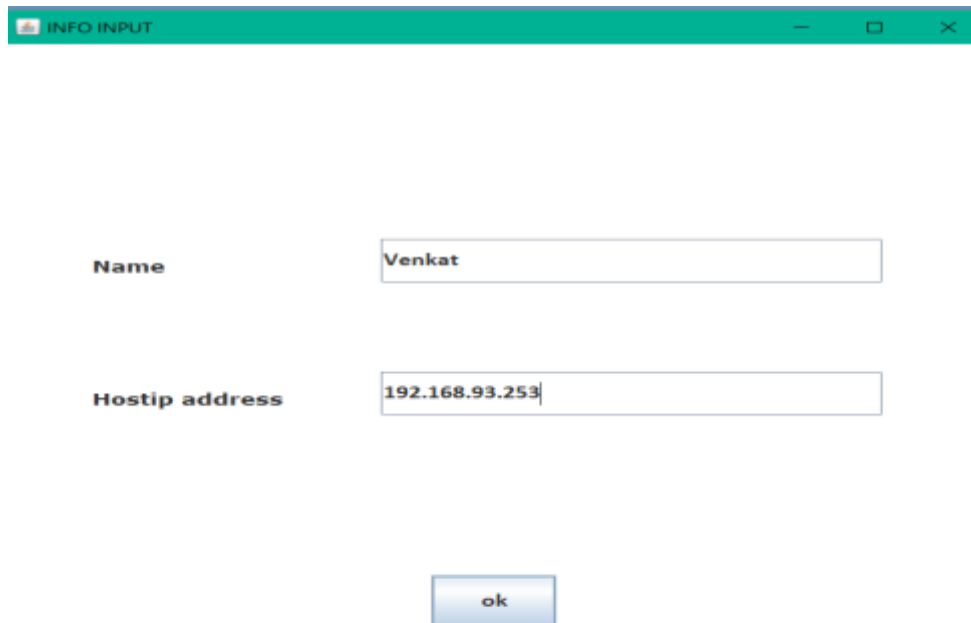
If you press yes you will be the host provided you are the only one and shows the frame like this below



A screenshot of a window titled "INFO INPUT" with a green header bar. The window contains two text input fields. The first field is labeled "Name" and is empty. The second field is labeled "chatroom name" and is also empty. Below the fields is a blue button with the text "ok".

Fig. 3.8.host entry pane.

Entry the fields and click ok



A screenshot of the same "INFO INPUT" window. The "Name" field now contains the text "Venkat". The "Hostip address" field (labeled as such in the image) now contains the IP address "192.168.93.253". The "ok" button remains at the bottom.

Fig. 3.9.host entry filled pane.

The application like this opens with host IP address displaying on the right top corner differentiating from the client applications

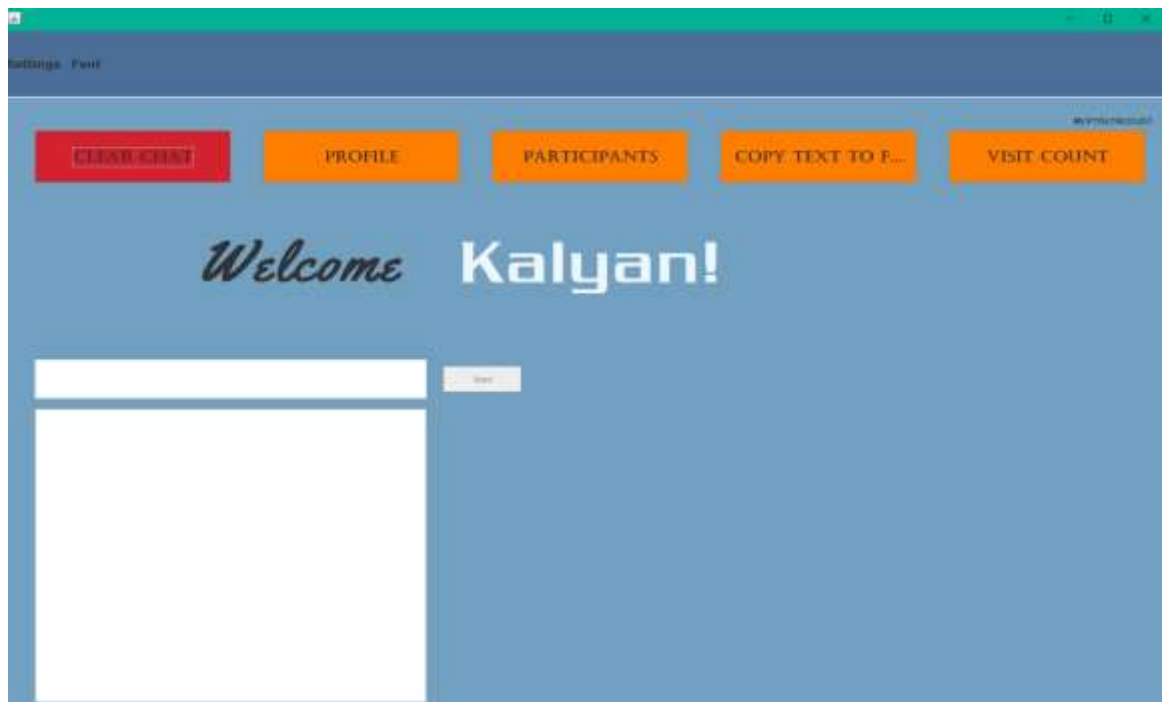


Fig. 3.10.host application.

If you have selected client mode then you will get a frame like this, enter the details and press OK button

A screenshot of a client entry pane titled "INFO INPUT". It contains two text input fields. The first field is labeled "Name" and contains the text "Venkat". The second field is labeled "Hostip address" and contains the text "192.168.93.253". Below these fields is a blue "ok" button.

Fig. 3.11.client entry pane.

On pressing the OK button you will get the application like this which contains the Chatroom name at the top right corner of the application

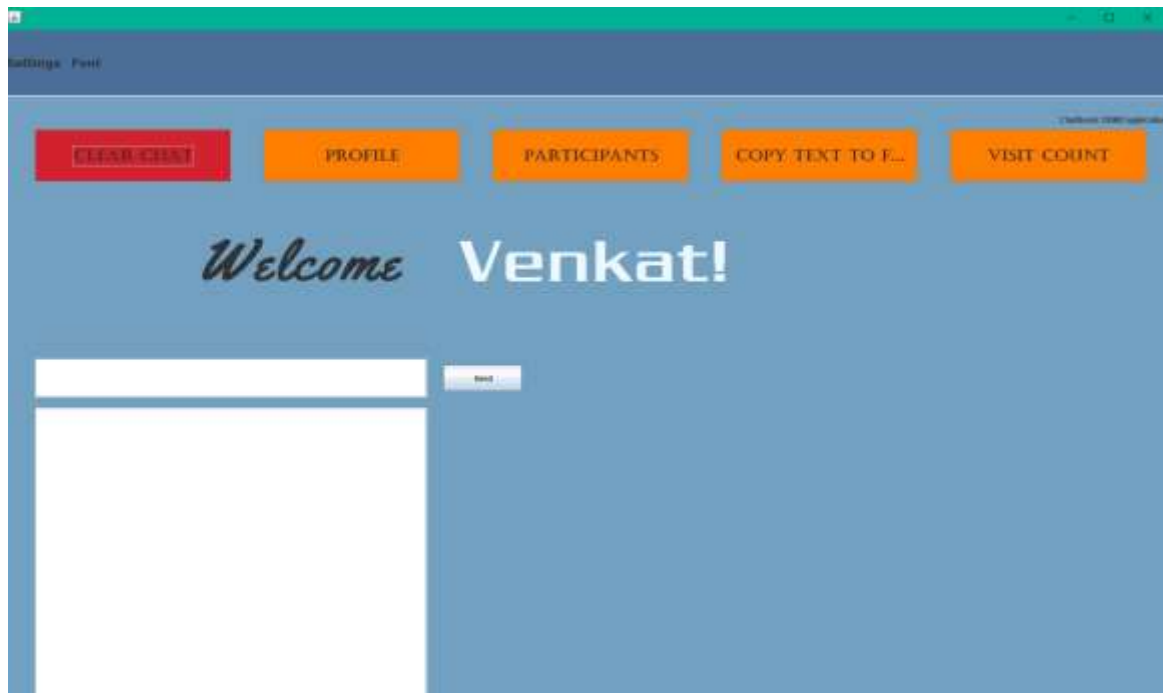


Fig. 3.12.client application 1.

Similar to the client Venkat create other clients as many you want for example here,



Fig. 3.13.client application 2.

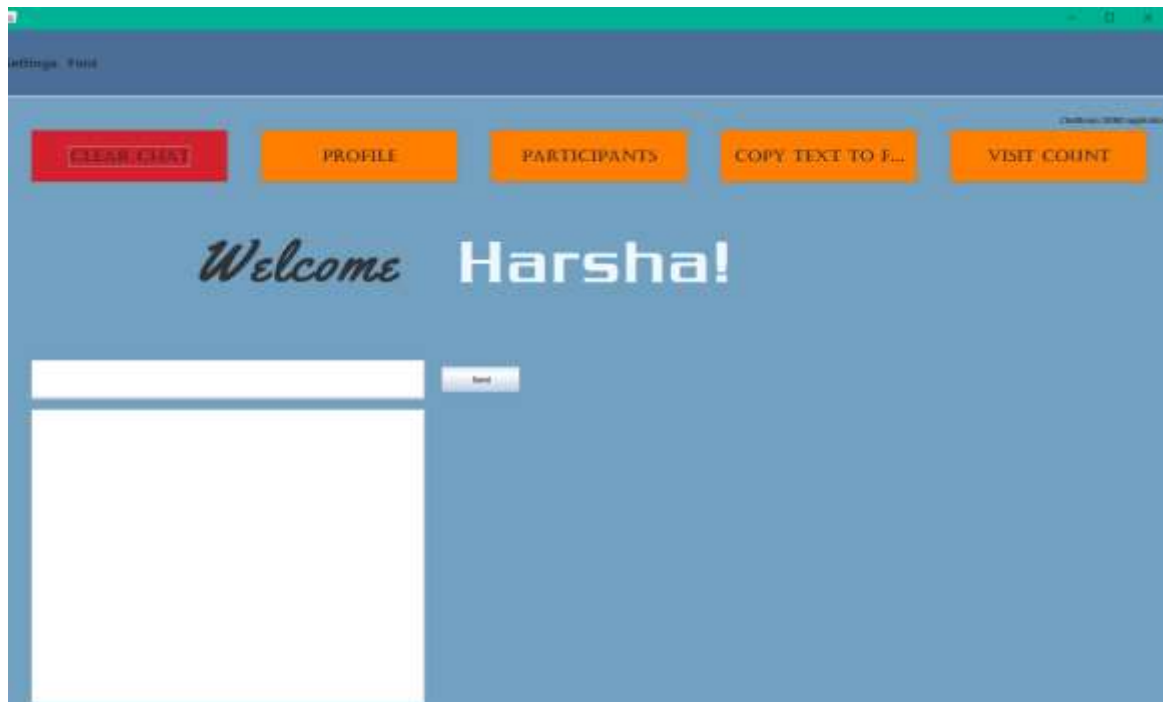


Fig. 3.14.client application 3.

The entry of the each client into the chat room will be notified in the text area like this

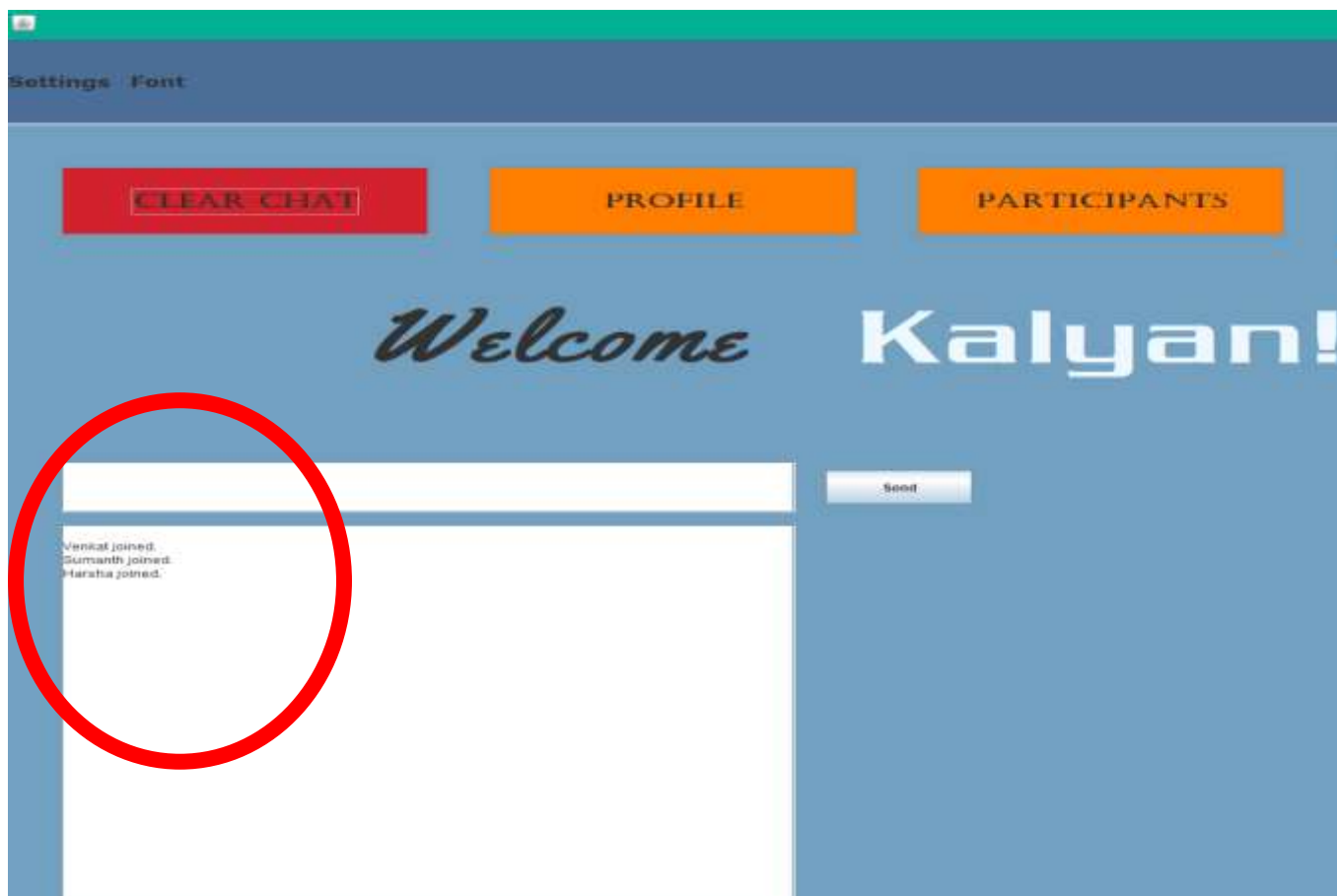


Fig. 3.15.joined display 1.

This notified information will be displayed only to the users who are in the chat room well advanced to the new user arrival for example user Sumanth will only be notified about the arrival of Harsha but not the previous ones

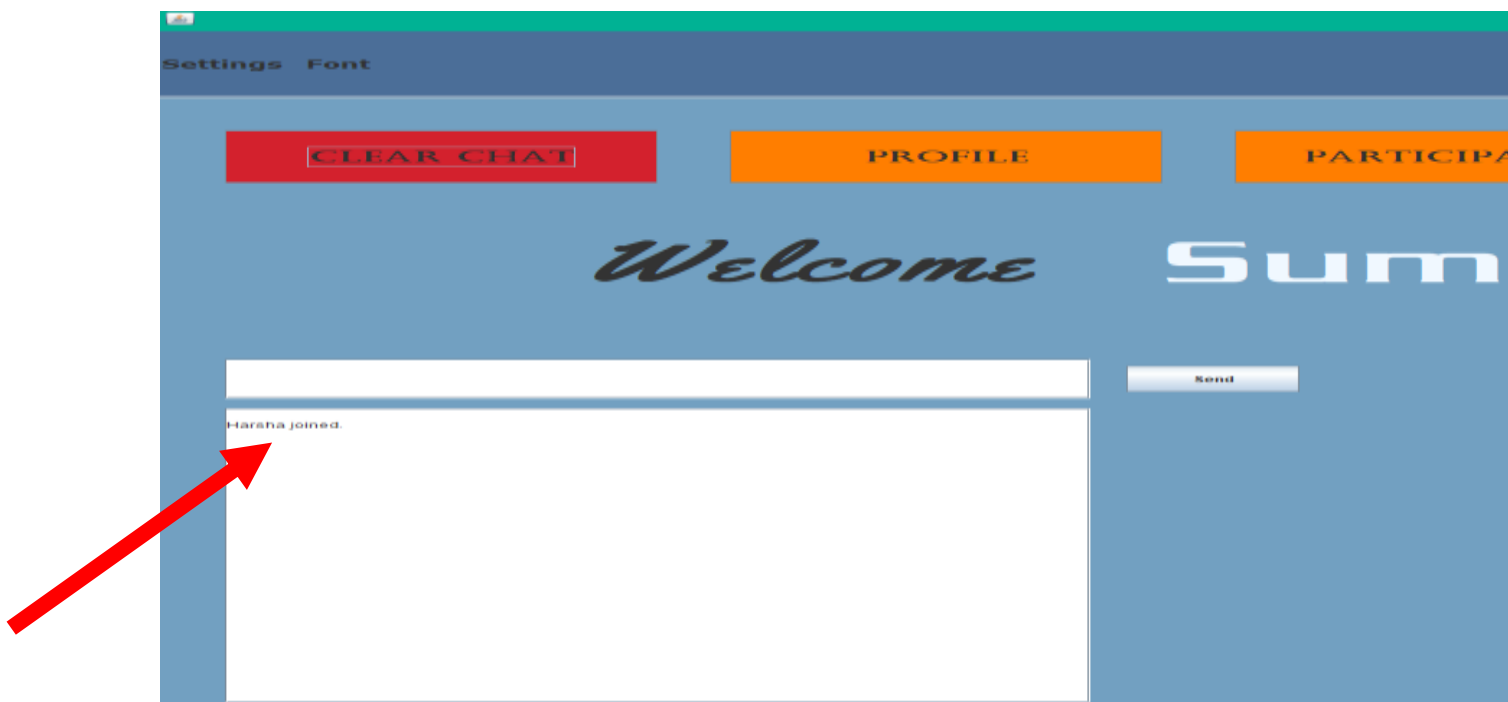


Fig. 3.16.joined display 2.

An one can send messages by typing in the text field ever user of the chatroom receives the message for example if harsha sends the message what are you doing?

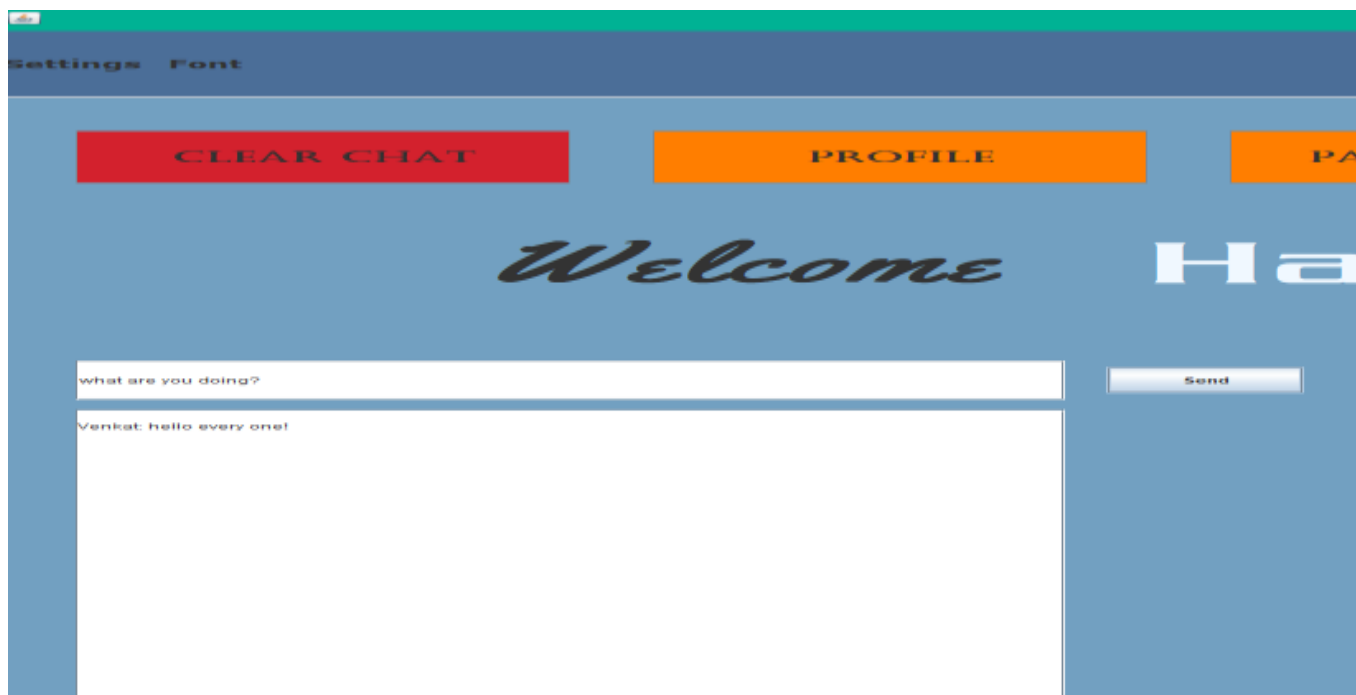


Fig. 3.17.send message.

It will be received by all other users of the chatroom

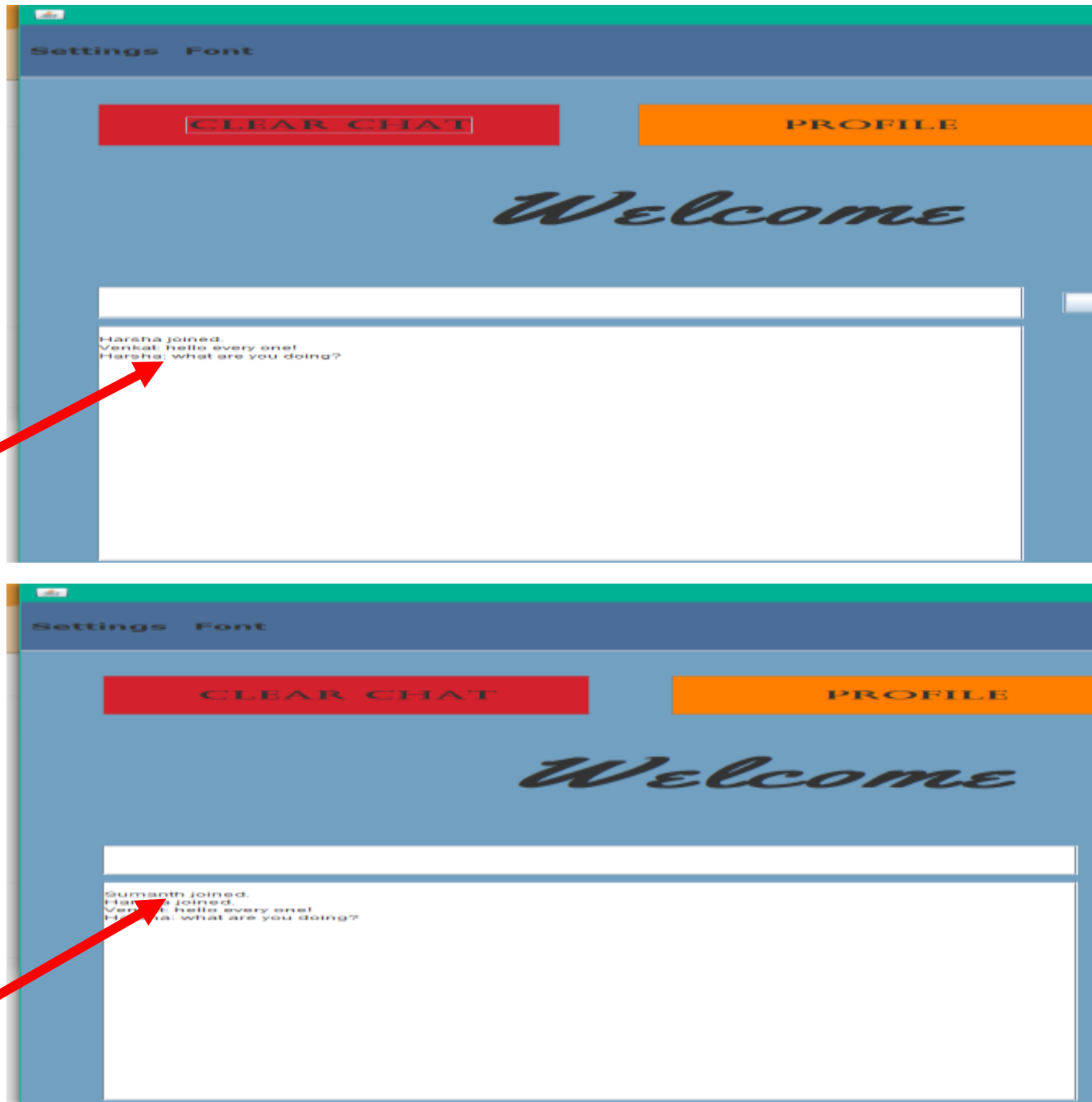


Fig. 3.18.receive message.

Not only this part, we will see the GUI options now on clicking the clear chat button this option pane displays for clarification

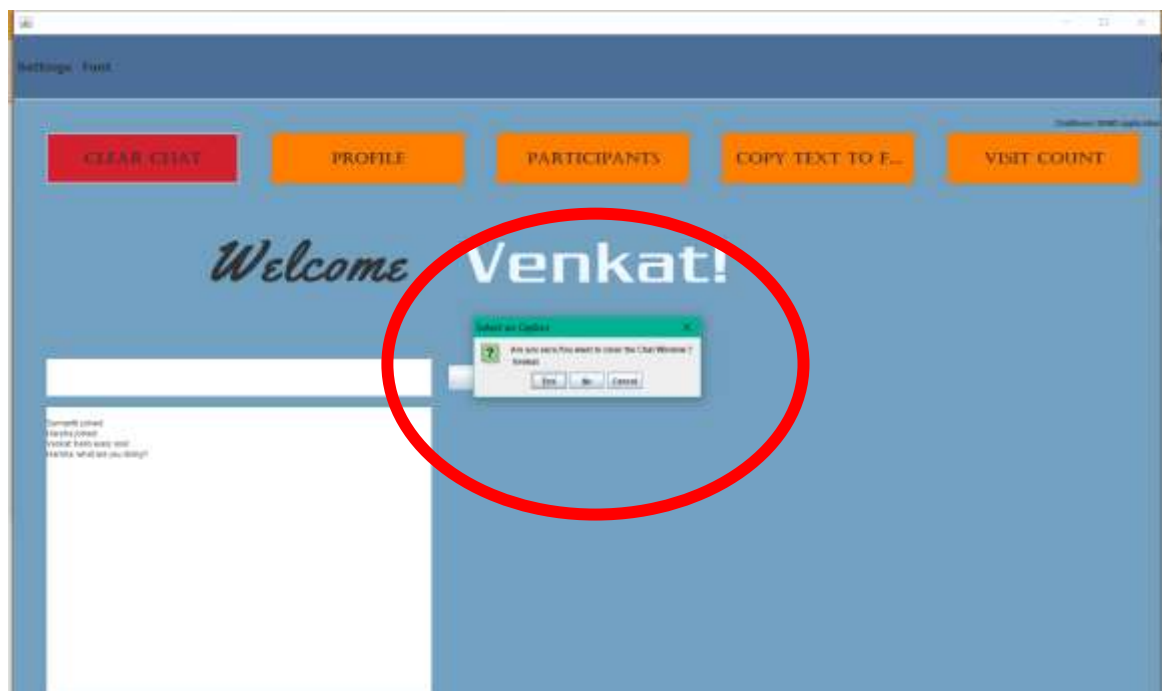


Fig. 3.19.clear button ask.

If you press yes the yes then text area gets cleared like this



Fig. 3.20.clear button result.

On clicking the profile button the information frame displays like this

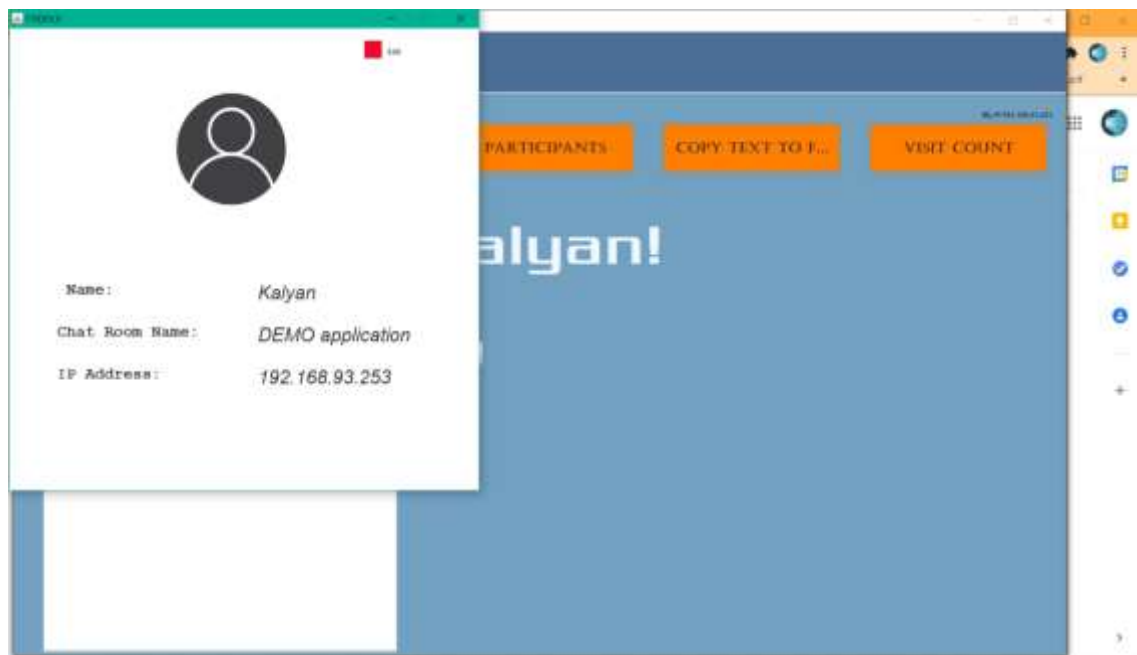


Fig. 3.21.profile button result.

You can change the photo of your profile if you want to by clicking the edit button on the top right corner and choosing the file from JFile chooser displayed with options, example of after changing it looks like this

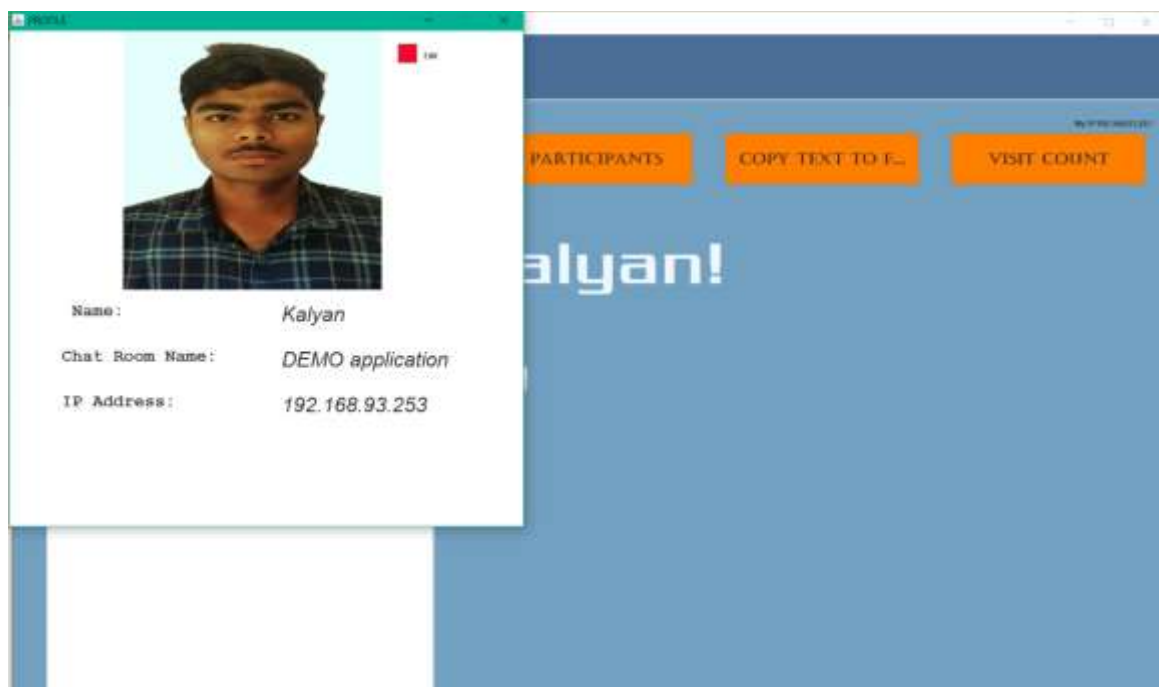


Fig. 3.22.profile button result modified.

Not only that on clicking the participants button you can see the list of all participants of the chat room in the table with their other information like IP address and port number. Additionally the user of the application will be separated from the rest by a free row and indicated as the owner in the braces.

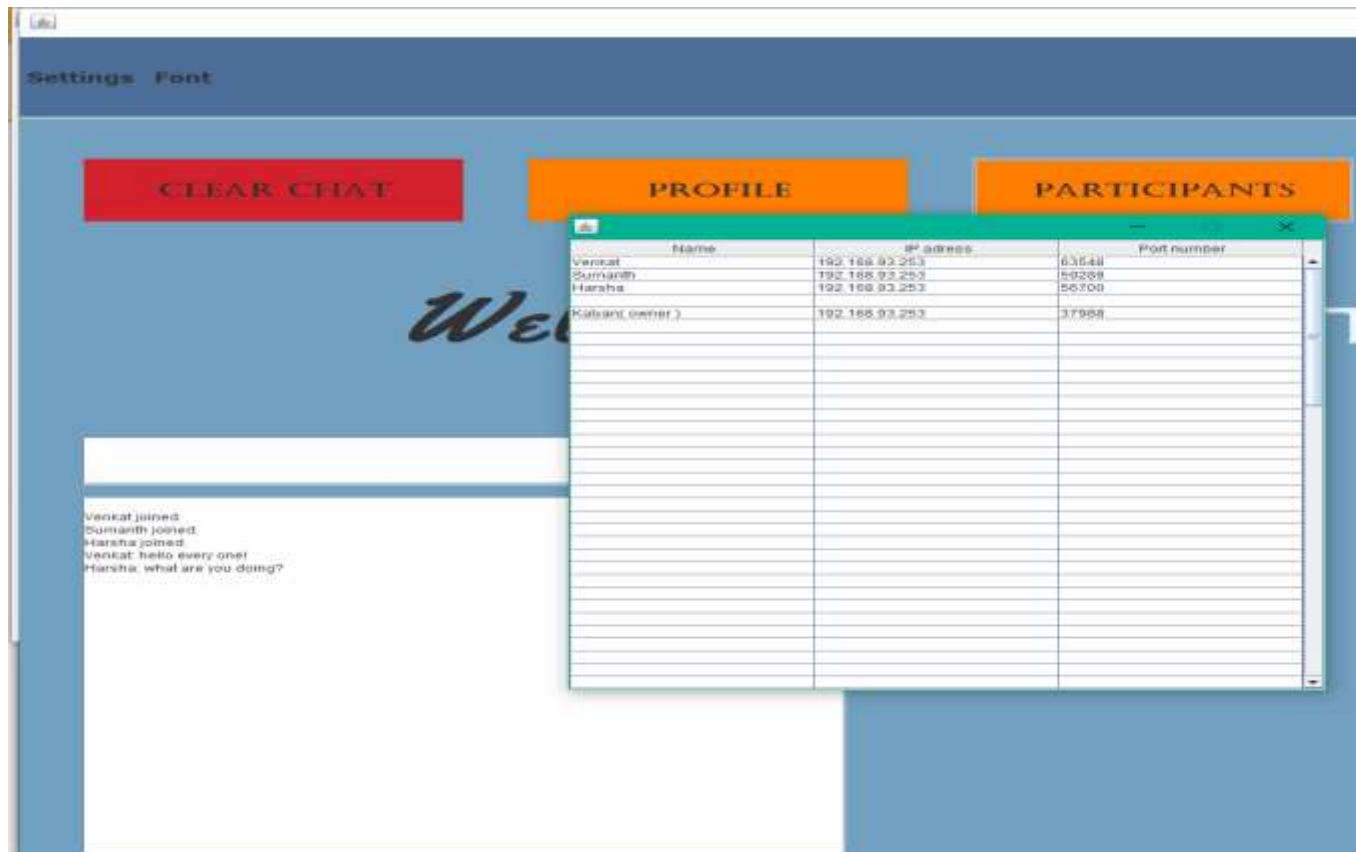


Fig. 3.23.participants button result.

You can also see the no. of visits made to the application so far on clicking visit count button



Fig. 3.24.count button result.

You can save the text in the text area in to a file on clicking the copy text to file buton

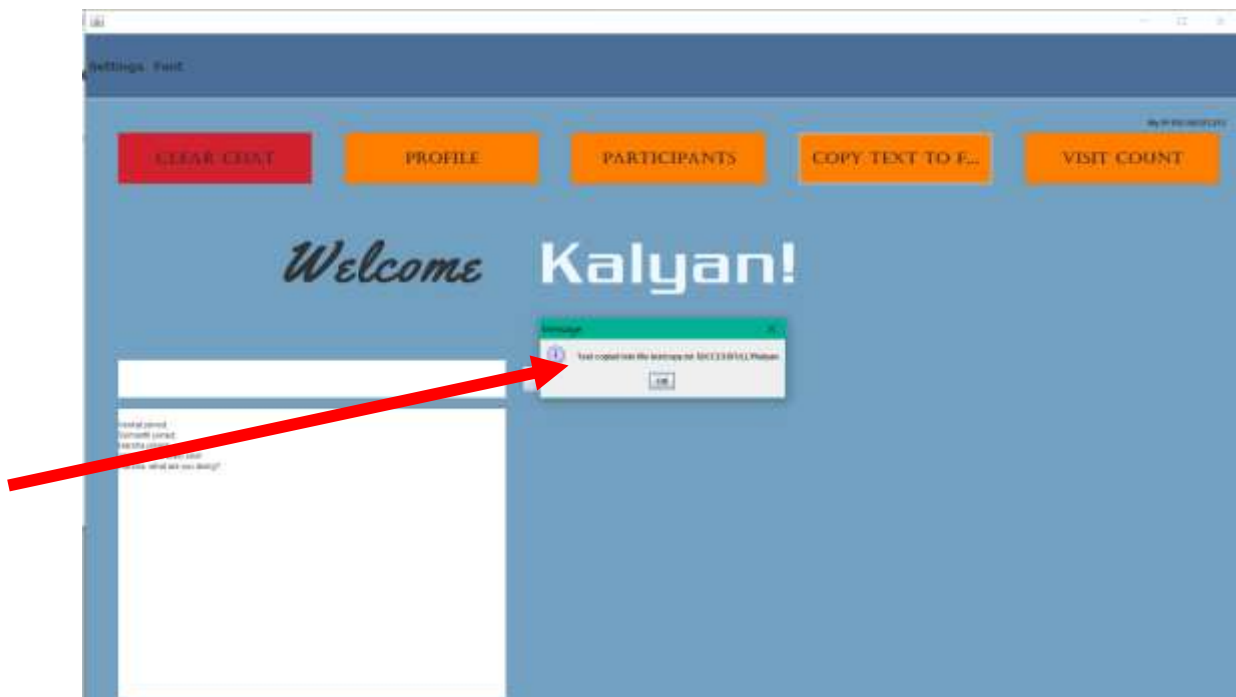


Fig. 3.25.copy text button result.

In J menu on clicking setting option you can change the frame colour and also the menu colour along with the option to close the application. The colours from the list shown can be selected to change the colour of the frame, menu etc.

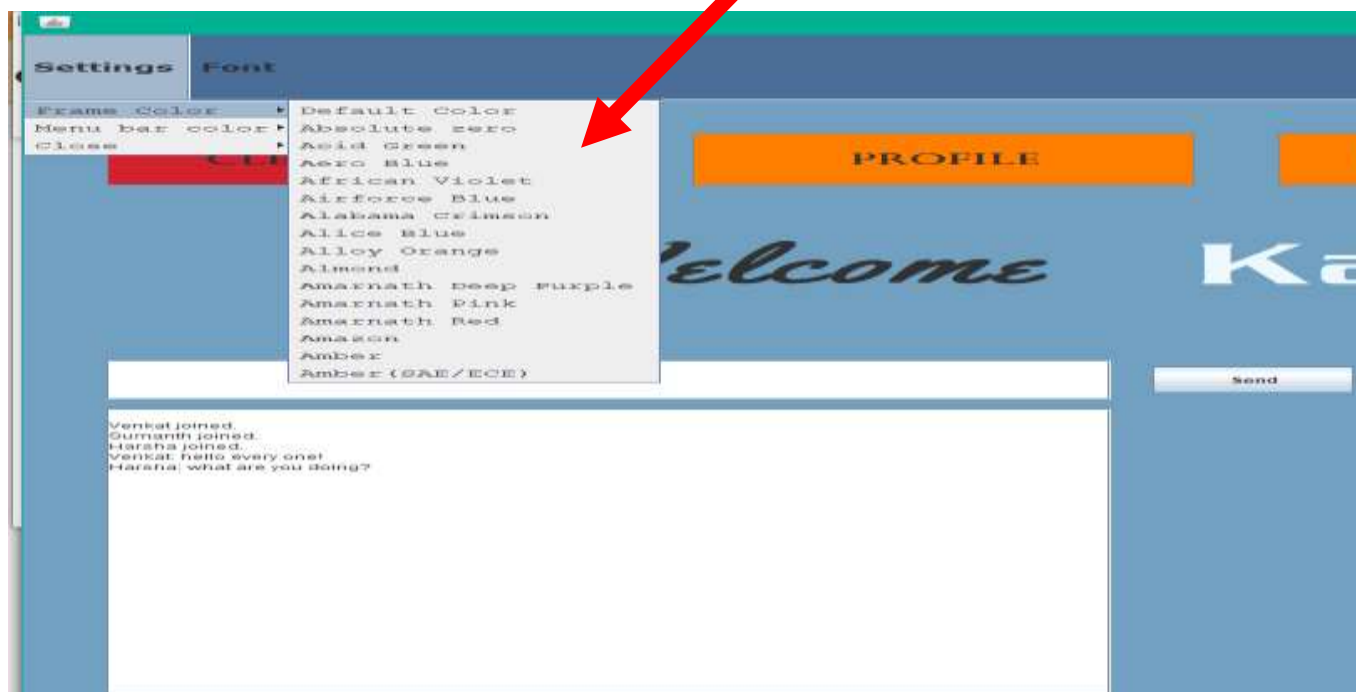


Fig. 3.26.settings menu.

Some of the examples of changing the frame colours



Fig. 3.27.frame colour change 1.



Fig. 3.28.frame colour change 2.

Some of the examples of changing the menu colour



Fig. 3.29.menu colour change 1.



Fig. 3.30.menu colour change 2.

The font button consists of three options to change the font of the text area you can change the font style, size and also colour of the text



Fig. 3.31.font menu.

Changing the font style examples



Fig. 3.32.font change .

Changing the font size example

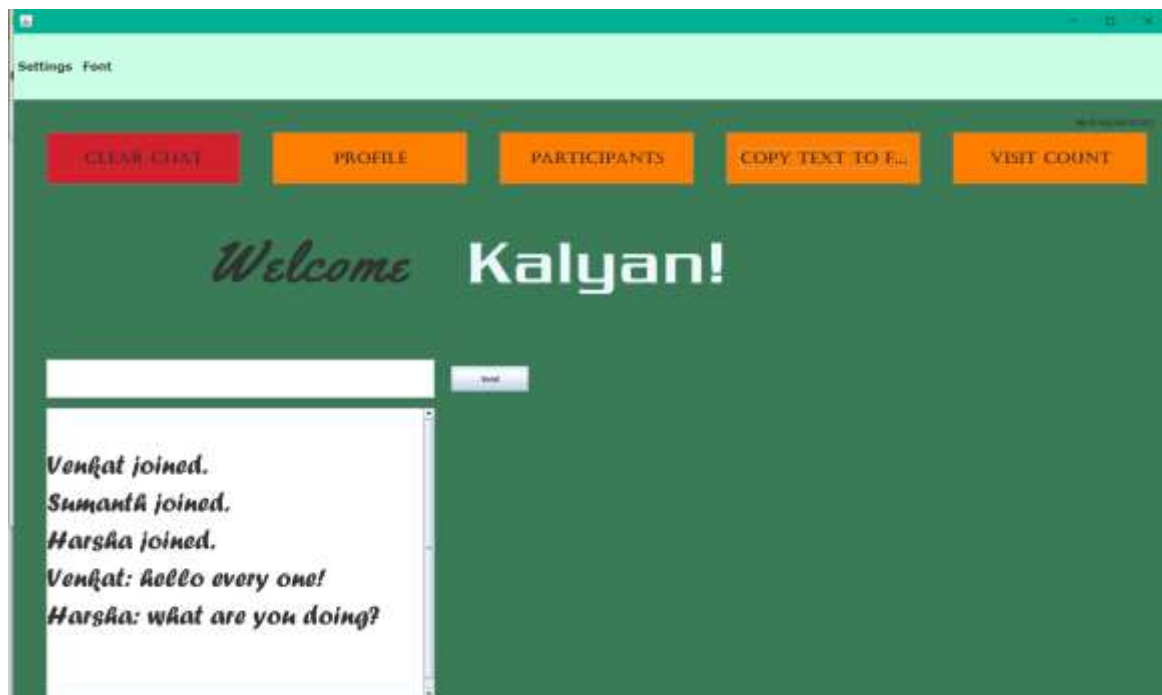


Fig. 3.33.font size change.

Changing the colour of the font example

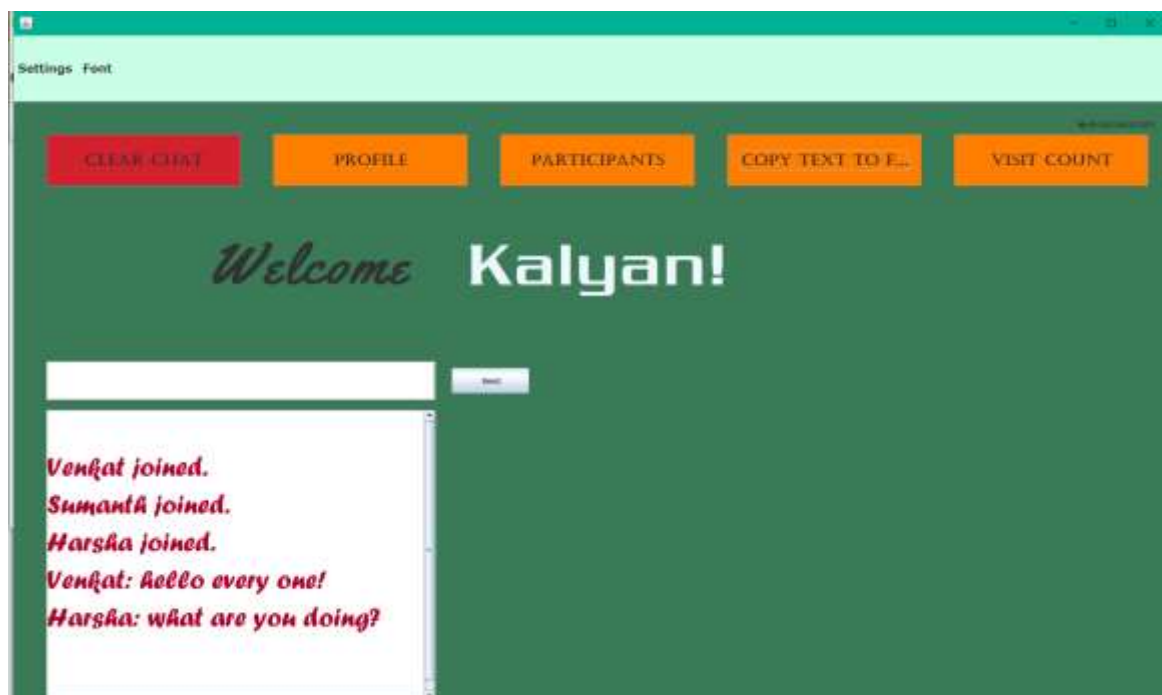


Fig. 3.34.font colour change.

CONCLUSION

The application created uses the good concepts like UDP, SMTP and AES encryption decryption concepts. And all these concepts are used in a well resource full manner to bring the best out of them. The Application is smooth running, easily portable and flexible with the system. The usage of GUI to create a good impression along with personalization for each user makes it unique and stand out from the one-other. Overall, the applications security maintained using the SMTP authentication, sending the messages among the several users using the UDP and maintaining the data integrity and protection through the encryption and decryption mechanisms makes the application good to use with belief and confidence. Here thereby can conclude that the chat room application created works good both ethically and technically.

FUTURE PLANS

On the successful implementation of this chat room application, I filled myself with the confidence of determination to work out these computer network concepts vastly and to develop a more sophisticated and robust application that can stand against any of the leading Networking applications of the Technological world.

I also would like to do the extension of this project level in to the mobile android and IOS devices in a way that the user data is given the at most protection along with the user-friendly usage to serve the people in all possible ways.

References

<https://ieeexplore.ieee.org/document/5486192/references#references>

Liu, J., Yan, G.-f.: Network Programming Technique and Its Realization Based on Socket. Journal of Jiangnan University (3) (2002)

Wang, F., Luo, J.-R.: The Research and Application of Multi-threaded Communications. Computer Engineering and Applications (16), 42–45 (2004)

ohnson, K (2000). Internet Email Protocols: A Developer's Guide. Addison-Wesley Professional

Hunt, C (2003). sendmail Cookbook. O'Reilly Media

"Biclique Cryptanalysis of the Full AES" (PDF). Archived from the original (PDF) on March 6, 2016. Retrieved May 1, 2019