

A
PROJECT REPORT
On
LAN CHAT & FILE SHARING APPLICATION

Submitted in partial fulfilment of the requirement for the award of the
Degree of
Bachelor of Engineering
In
Information Technology
By

MIRZA ILIYAZ BAIG (1604-18-737-308)
MOHD AFROZ (1604-18-737-091)
SHAIK ASHFAQ AHMED (1604-18-737-095)

Under the Guidance
Of
MOHD. ABDUL RASHEED
(ASSISTANT PROFESSOR)



DEPARTMENT OF INFORMATION TECHNOLOGY
MUFFAKHAM JAH COLLEGE OF ENGINEERING AND
TECHNOLOGY

(Affiliated to Osmania University)
Banjara Hills, Hyderabad 500034
2020 - 2021

MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to Osmania University)
Banjara Hills, Hyderabad - 500034

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project report entitled **LAN Chat & File Sharing Application** being submitted by **Mirza Iliyaz Baig, Mohd Afroz, Shaik Ashfaq Ahmed**, bearing Roll No. **1604-18-737-308, 1604-18-737-091, 1604-18-737-095** respectively, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Engineering in Information Technology to the Osmania University is a record of bonafide work carried out by him under my guidance and supervision.

Dr. Mousmi Ajay Chaurasia
(Head of the Department)

Mohd. Abdul Rasheed
(Assistant Professor)

ACKNOWLEDGEMENT

In the accomplishment of our project LAN Chat & File Sharing Application, We would like to express our sincere thanks to **Mohd. Abdul Rasheed**, Assistant Professor, of Information Technology, Muffakham Jah College of Engineering & Technology for his valuable guidance and support in completing my project. Without his support and suggestions, this project would not have been completed.

We would also like to express our gratitude to **Dr. Mousmi Ajay Chaurasia**, Head of the Department of Information Technology, Muffakham Jah College of Engineering & Technology for giving me this great opportunity to undertake this project.

We are indebted to **Muffakham Jah College of Engineering & Technology** for providing us the healthy, supportive & learning environment to drive us to achieve our ambitions and goals. Theoretical knowledge is of no importance if one doesn't know the way of its implementation. We are thankful to our institute that provided us an opportunity to apply our theoretical knowledge through the project. We feel obliged in submitting this project as part of our curriculum.

Finally, we would like to express our gratitude to one and all that have helped us in successfully completing this project. Furthermore I would like to thank my family and friends for their moral support and encouragement.

Mirza Iliyaz Baig (1604-18-737-308)
Mohd Afroz (1604-18-737-091)
Shaik Ashfaq Ahmed (1604-18-737-095)

**Department of Information Technology,
Muffakham Jah College of Engineering & Technology.**

LAN CHAT & FILE SHARING APPLICATION

ABSTRACT

LAN Chat & File Sharing Application is one of easiest way to chat with your friends & share files to each other through LAN (Local Area Network). No internet connection is needed. The only thing which requires is server IP address and you will be able to connect to others members through LAN. It can help you to chat with your friends even you both do not have internet connection. As it is based on LAN. Local area network (LAN) which connect different client to each other and also client to main server. So I have used the same concept here, we are connecting two or more client to the server with each other and by providing the IP address we can talk with each other. On the other hand file sharing application is also implemented where a user can upload files on a server specified path. We are providing the path to the directory where the uploaded files are stored for the user who wants to access particular file.

Two sockets are created at the client side and the server side. The client connects to the server through its IP address and port number. They must share the same port number for them to communicate. The client and the server both communicate through a stream of bytes written to the socket. Basically, the message sent by one client into a socket and passes it to another client on the receiving side. If it is group chatting, a central socket will be used to collect the message and then it will be broadcast to all clients that are active.

INDEX

CONTENTS	Page No.
1. INTRODUCTION	01
1.1 Project Details	01
1.2 Purpose	01
1.3 Scope	01
1.4 Objective	01
2. SYSTEM REQUIREMENTS	02
2.1 Hardware Requirement	02
2.2 Software Requirement	02
3. TECHNOLOGY AND LITERATURE REVIEW	03
3.1 Why JAVA?	03
3.2 Eclipse IDE	04
3.2.1 Best Support for Latest Java Technologies	04
3.2.2 Fast & Smart Code Editing	04
3.2.3 Easy & Efficient Project Management	04
3.2.4 Write Bug Free Code	04
3.3 Network	05
3.3.1 IP Address	06
3.3.2 Service Port	06
3.3.3 Sockets	06
4. REQUIREMENTS ANALYSIS	07
4.1 Existing System	07
4.2 Problems And Weakness Of Existing System	07
4.3 Proposed System	07
4.4 Features of New System	07
4.5 User Characteristics	08
4.6 Packages Used	08
4.7 Assumptions And Dependencies	08

5. SOFTWARE DESIGN	09
5.1 Use Case Diagram	09
5.2 Activity Diagrams	10
5.3 Sequence Diagram	11
6. CODING	12
6.1 Server.java	12
6.2 Client.java	15
7. OUTPUT SCREENS	22
7.1 Running Server & Client	22
7.2 Login Screen	23
7.3 Chat Screen	24
7.4 File Selector Screen	25
7.5 Logout	26
7.6 Chat History	27
8. CONCLUSIONS AND DISCUSSION	28
9. REFERENCES	29

1. INTRODUCTION

1.1 Project Details

The project entitled “LAN-Chat & File Sharing Application” is a Chatting Application that enables different type of users like Students and Faculties to do interaction with each other and enables them to share important information with this application. Main aim of this application is to provide an easy way to do conversation and announce any event to all or specific user.

1.2 Purpose

In Today’s world the important thing which is required is time. Now every universities and colleges has so many resources available for everyone. But to manage them and to manage student-faculty conversation, it takes so much efforts if done manually.

This project is developed mainly to reduce that efforts by making it automatically as much as possible. Which reduces the efforts.

1.3 Scope

This application would be used by any well-established LAN network and server which is created by us, has to be deployed on the local server of the LAN network.

1.4 Objective

- To secure the data and information by sending it through LAN network.
- Saves up time for data processing.
- Secure connection for interaction between clients.
- Speed transferring of files.
- Using this application user can do:
 - Can do Chatting
 - Can send file

2. SYSTEM REQUIREMENTS

2.1 Hardware Requirement

- Processor : Intel Pentium IV 3.5 GHz or above.
- Ram : 256 MB.
- Hard Disk : 40 GB.
- Monitor : 14" Colour Monitor.
- Mouse : Optical Mouse.
- Keyboard : 104keys Keyboard

2.2 Software Requirement

- Operating System : Windows XP/2000/vista/7/8/10
- Coding Language : Java
- IDE : Eclipse IDE
- Development Kit : JDK 16.0.2
- Documentation : MS Office

3. TECHNOLOGY AND LITERATURE REVIEW

3.1 Why JAVA?

Java has significant advantages over other languages and environments that make it suitable for just about any programming task.

The advantages of Java are as follows:

- Java is easy to learn.

Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.

- Java is object-oriented.

This allows you to create modular programs and reusable code. Everything in Java is an object which takes care of both data and behaviour. Java uses object-oriented concepts like object, class, inheritance, encapsulation, polymorphism and abstraction.

- Java is platform-independent.

One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels. Because of Java's robustness, ease of use, cross-platform capabilities and security features, it has become a language of choice for providing worldwide Internet solutions.

- Multi-Threaded

Java uses a multi-threaded environment in which a bigger task can be converted into various threads and run separately. The main advantage of multi-threading is that we need not provide memory to every running thread.

3.2 Eclipse IDE

3.2.1 Best Support for Latest Java Technologies

Eclipse IDE provides first-class comprehensive support for the newest Java technologies and latest Java specification enhancements before other IDEs. It is the first free IDE providing support for JDK 8 previews, JDK 7, Java EE 7 including its related HTML5 enhancements, and JavaFX 2. With its constantly improving Java Editor, many rich features and an extensive range of tools, templates and samples, Eclipse IDE sets the standard for developing with cutting edge technologies out of the box.

3.2.2 Fast & Smart Code Editing

An IDE is much more than a text editor. The Eclipse Editor indents lines, matches words and brackets, and highlights source code syntactically and semantically. It also provides code templates, coding tips, and refactoring tools.

3.2.3 Easy & Efficient Project Management

Keeping a clear overview of large applications, with thousands of folders and files, and millions of lines of code, is a daunting task. Eclipse IDE provides different views of your data, from multiple project windows to helpful tools for setting up your applications and managing them efficiently, letting you drill down into your data quickly and easily.

3.2.4 Write Bug Free Code

The cost of buggy code increases the longer it remains unfixed. Eclipse provides static analysis tools, especially integration with the widely used Find Bugs tool, for identifying and fixing common problems in Java code. In addition, the Eclipse Debugger lets you place breakpoints in your source code, add field watches, step through your code, run into methods, take snapshots and monitor execution as it occurs.

3.3 Network

A network includes a group of computers connected by a physical link allowing data to be exchanged between them. A local area network on LAN is a network of computers in close physical proximity, usually a single building, but can be a group of adjacent buildings. Over the last decades LANs have become an important component of the computer workplace.

Client/server network was used to implement the LCS because of the central administration, scalability- any component or client can be upgraded when required, flexibility – new technology can be integrated into the network should the client increase. Table 1 below shows the comparison between peer to peer network and client/server network.

Table 1: Comparison between peer to peer network & the Client/Server network architecture

	Peer to Peer Network	Client/Server network
Hardware Cost	It needs no high-end server as the resources are distributed over all clients which reduce cost.	A dedicated computer server (hardware) that distributes resources is needed.
Easy Setup	It is easy to setup mainly if the computers are less than fifty (50).	It is difficult to setup.
Network Operating system	There is no required network operating.	Network operating system is required.
Failure	It can accommodate failure i.e. if one or more Computers (Clients) fail the others can still be up.	It cannot accommodate failure if the server fails.
Security	It has security deficiency as clients' administration is not guaranteed.	Very secure because server administration is guaranteed.
Performance	It performs less	Performs very good
Backup	It has decentralized backup that is difficult to access.	It has a centralized data backup with ease of access.

3.3.1 IP Address

An IP address uniquely identifies any computer connected to a network. This address is made up of 32 bits divided into 4 four bytes. But since the number of connected computers is too large and since it is difficult to remember all their IP addresses, the Domain Name Service (DNS) was designed. It has the job of transforming the unique computer names (host name) into an IP address. Therefore, whenever in our project we run the client application and enter the host name, this means that we are writing the IP address of the remote computer we want to connect to indirectly. In general, TCP/IP is a set of protocols developed to allow cooperating computers to share resources across the network.

3.3.2 Service Port

Till now, we have seen that TCP/IP forms the backbone for communication between computers, but do you know how these computers speak to each other?

The answer is Ports. A port is a special location in the computer's memory that exists when two computers are communicating via TCP/IP. Application uses a port number to communicate and the sending and receiving computers use this same port to exchange data. To make the job of communication easier, some port numbers have been standardized, Ex, (www Port 80, Ftp Port 20, 21, Etc.). In our application we are using custom port = 7777.

3.3.3 Sockets

The world is defining itself as a largely Intel-processor, windows-based set of desktops communicating with back end servers of various types. Hardware and software technology advances are pushing PC's into the role of everywhere communications devices. For software applications to take advantage of increasingly sophisticated and feature-rich communications technology, they require an Application Programming Interface (API) which provides a simple and uniform access to this technology.

4. REQUIREMENTS ANALYSIS

Requirement Analysis is done to understand the problem the software system is to solve. The emphasis in requirements analysis is on identifying what is needed from the system, but not how the system will achieve the goals.

4.1 Existing System

- Manually work done in most of the colleges and universities.
- If you can see current system, there is not secure way to communicate with each other.
- Also, in current systems there is no restriction over chatting (live messaging).

4.2 Problems and Weakness of Existing System

- The current system is not that efficient. The user has to have internet connection whenever he wants to access the application within college. User can't get access to resources directly. First they have to register themselves to admin, then they would get their passwords for chatting.
- GUI is not improved.

4.3 Proposed System

It is a client-server system where all clients are connected to the server via LAN. This chat application can also be used for group discussions. The client can only connect when the SERVER IP address is known, we use this application via LAN.

4.4 Features of New System

- Easy to use
- Reliable and accurate
- Provides functionalities of sending documents and messages to faculties or students, view old messages, view received files and provide ability to download it, edit profile and basic functionalities.
- Secure, as it sends passwords to authorized email-ids only.
- User friendly GUI.

4.5 User Characteristics

- User can do private chat,
- User can do group chat,
- User can share files,
- See old messages

4.6 Packages Used

1. The default package to build any program in Java is **lang** that is used in developing this system
2. The API to build Graphical User Interface in Java is provided using the packages **awt** and swing. This system uses both the packages to provide GUI based forms.
3. To provide the communication over the network, the package **net** is used to develop the Server and Clients.
4. The **util** package in Java is an enhancement of the existing classes. This package is also used in this system for declaring arrays (Vector), and to manipulate the strings.
5. The streams that are used to send and receive the data over network are also utilized in this system, which are in the package of **io**.

4.7 ASSUMPTIONS AND DEPENDENCIES

Assumptions are described below:-

- User has sufficient privileges to access LAN.
- Server is running correctly.
- Server is running in the same network as in users are logged in.

Dependencies are described as below:-

- This system is dependent upon the IP address and port number. If the port number is not matched with the server's port number then user will not be able to connect.
- This application depends on the server. So if server works correctly then only this application can run in proper way.

5. SOFTWARE DESIGN

Design is concerned with the mapping of objects in the problem space into objects in the solution space and creating an overall structure of the system.

The Object Oriented Design Methodology involves the following steps:

1. Review of Objects created in the analysis phase
2. Specification of class dependencies
3. Design of classes
4. Identifying the class dependencies

The best approach for Object Oriented Design is Unified Modeling Language (UML). The UML model describes what a system is supposed to do. It doesn't tell how important the system. Various tools are used to present multiple views of the system. The UML enables system builders that create blueprints that capture their visions in a standard, easy-to-understand way and communicate them to others.

5.1 Use case & Scenario for the need to connect to the Server.

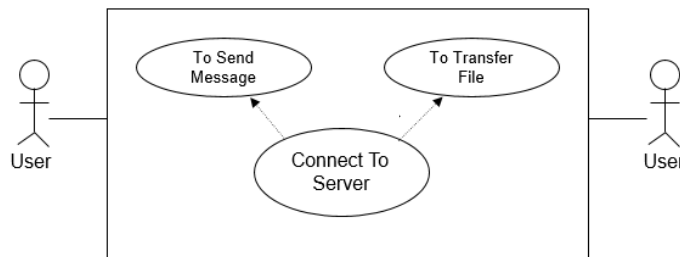


Fig-5.1: Use Case

Steps:

1. The User connects to the server
2. The User sends a message or file
3. Exit

The Relationship among Server and Clients can be represented by the following object diagram.

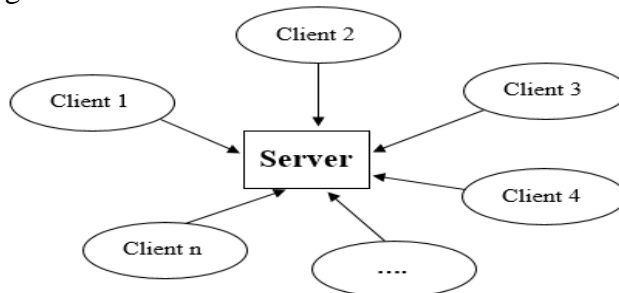


Fig-5.2: Object Diagram

5.2 Activity Diagrams

Activity Diagram shows the overall functionality of the system.

- Activity Diagram for Connecting to the Server.

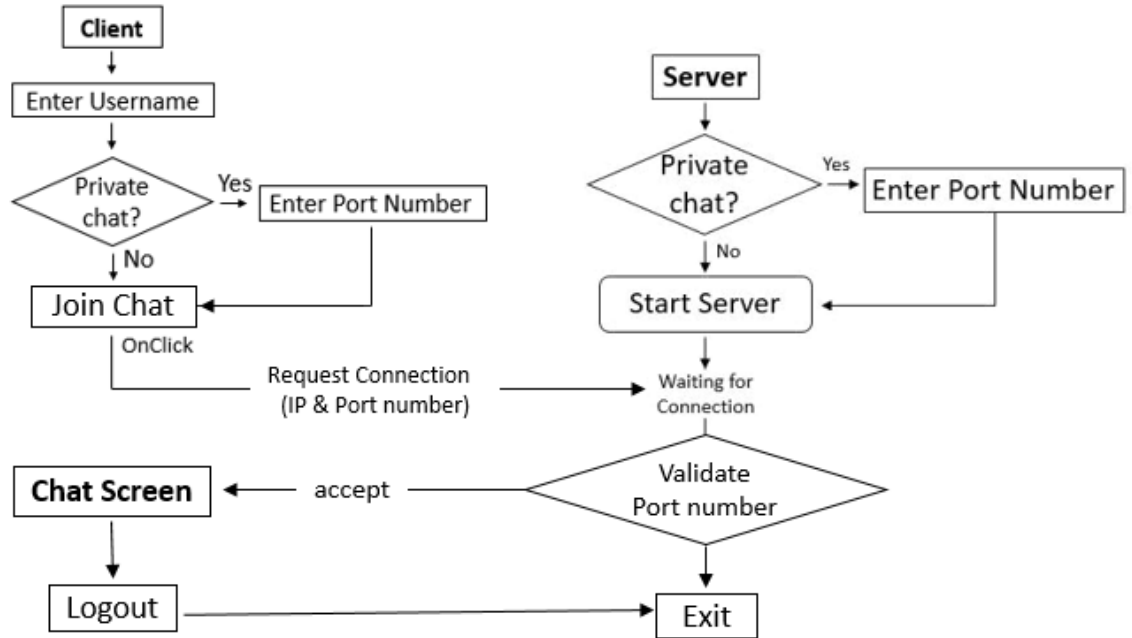


Fig-5.3: Activity Diagram

- Activity Diagram for Sending Message.

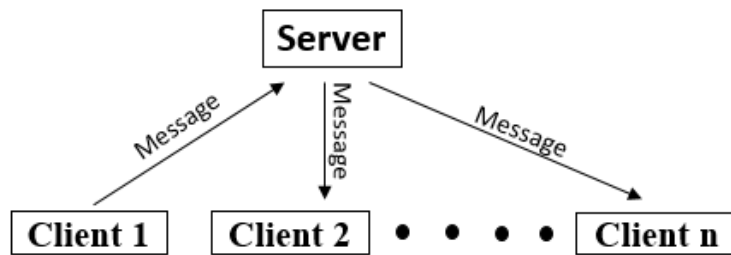


Fig-5.4: Activity Diagram of sending message

- Activity Diagram for Sending File.

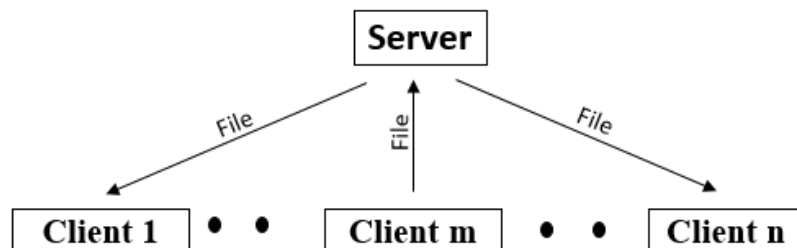


Fig-5.5: Activity Diagram of sending file

5.3 Sequence Diagrams

Sequence diagrams are used to represent the sequence of events basing on the time, i.e. these show the time-based dynamics of the interaction.

- Sequence Diagram for Clients that connect to the Server.

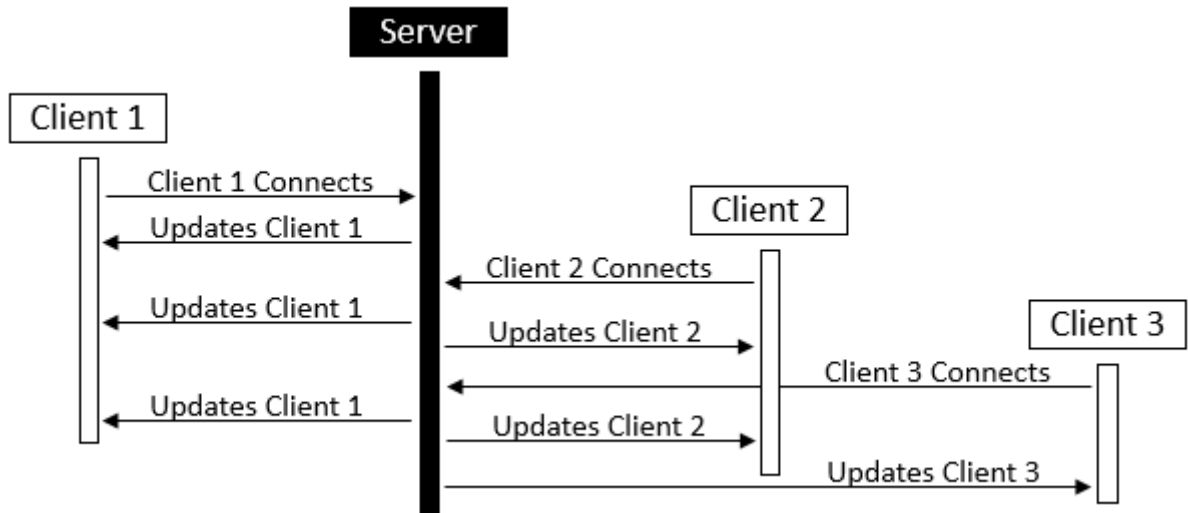


Fig-5.6: Sequence Diagram for clients that connects to server

- Sequence Diagram for sending message and transferring a file.

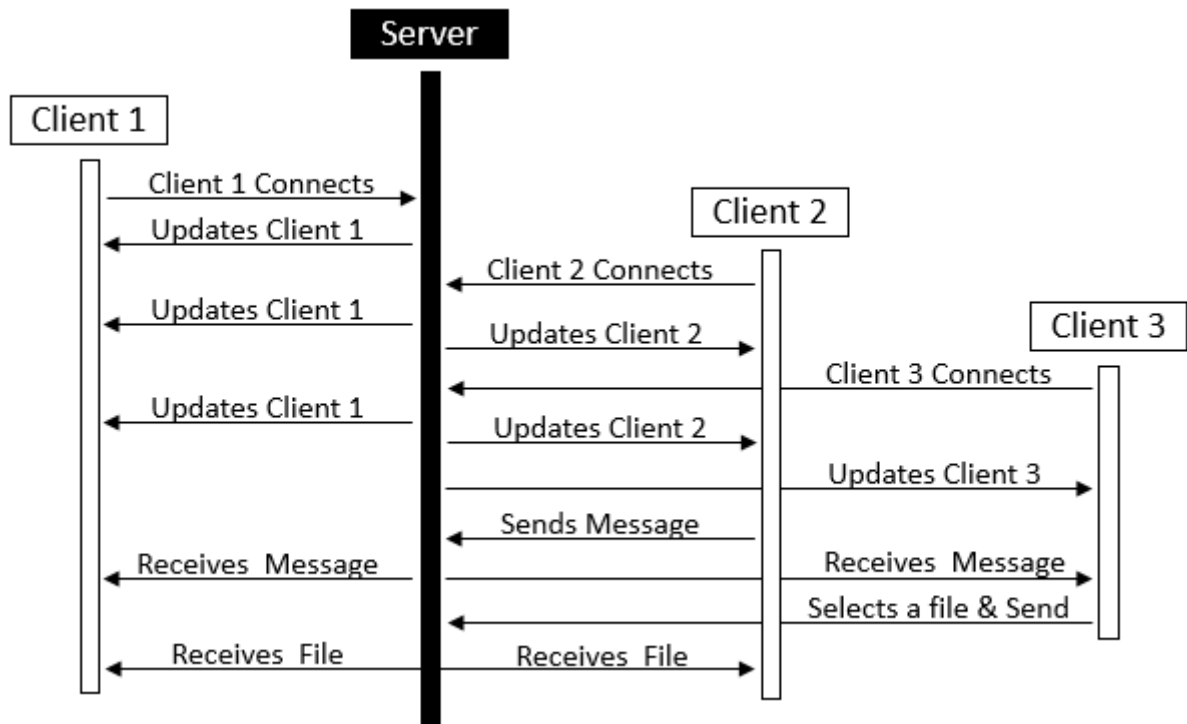


Fig-5.6: Sequence Diagram of sending message or file

6. CODING

6.1 Server.java

```
import java.util.Scanner;
import java.io.*;
import java.net.*;
import java.time.format.DateTimeFormatter;
import java.time.LocalDateTime;

public class server {
    static ServerSocket ss;
    static DataInputStream dis;
    static DataOutputStream dos;
    static Scanner in = new Scanner(System.in);
    public static int limit = 10;
    public static user users[] = new user[limit];
    public static int totalClientsOnline = 0;
    static String FILE_TO_RECEIVE;
    static int port;

    public static void main(String args[]) throws Exception {
        try {
            System.out.print("Private chat? [y/n]: ");
            char wc = in.next().charAt(0);
            if(wc == 'n') port = 7777;
            else { System.out.print("Port Number: ");
                port = in.nextInt();
            }
            ss = new ServerSocket(port);
            System.out.println("Server Started...");
            for(int i=0;i<limit;i++) {
                users[i] = new user(i+1,ss.accept());
            }
        } catch(Exception e) { System.out.println("Exception caught: "+e); }
    }

    public void MessageHistory(String chat) {
        try {
            DateTimeFormatter df = DateTimeFormatter.ofPattern("dd-MM-yyyy");
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd_HH:mm:ss");
            LocalDateTime now = LocalDateTime.now();
            FileWriter fw = new FileWriter("ChatHistory/"+df.format(now)+" Chat History.txt", true);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write(dtf.format(now)+" > "+chat+"\n");
            bw.close();
        } catch(Exception e){ System.out.println(e); }
    }
}
```

```

public void sendMessageToAll(String msg) {
    for(int c=0;c<totalClientsOnline;c++) {
        try {
            users[c].sendMessage(msg);
        } catch(Exception e){ }
    }
}

public void receiveAndSendFileToAll(Socket uSocket, int userID) {
    String workingDir = System.getProperty("user.dir");
    String FILE_TO_RECEIVED =workingDir +"\\Files\\"+FILE_TO_RECEIVE;
    int FILE_SIZE = 131000;
    int bytesRead;
    int current = 0;

    FileOutputStream fos = null;
    BufferedOutputStream bos = null;
    byte [] mybytearray = new byte [FILE_SIZE];

    try {
        InputStream is = uSocket.getInputStream();
        fos = new FileOutputStream(FILE_TO_RECEIVED);
        bos = new BufferedOutputStream(fos);
        bytesRead = is.read(mybytearray,0,mybytearray.length);
        current = bytesRead;
        bos.write(mybytearray, 0 , current);
        bos.flush();
        bos.close();
        System.out.println("File " + FILE_TO_RECEIVED+ " downloaded (" + current
                                + " bytes read)");
        for(int c=0;c<totalClientsOnline;c++) {
            if(c != userID-1) users[c].sendFile(FILE_TO_RECEIVE);
        }
    } catch(Exception e) { System.out.println("Exception caught in receiveFile()."); }
}

class user extends Thread {
    server tirth = new server();
    int userID;
    public Socket userSocket;
    public DataInputStream userDIS;
    public DataOutputStream userDOS;
    public Thread t;
    OutputStream os;
    public String secretCode = "46511231dsfdsfdsd#@$#$$@^$%#@*$#^";
}

```

```

public user(int id,Socket a) {
    try {
        userID = id;
        userSocket = a;
        userDIS = new DataInputStream(userSocket.getInputStream());
        userDOS = new DataOutputStream(userSocket.getOutputStream());
        tirth.totalClientsOnline++;
        System.out.println("Client connected with id"+userID);
        userDOS.writeUTF(secretCode+"User"+userID);
        t = new Thread(this);
        t.start();
    } catch(Exception e) { System.out.println("Exception caught in constructor."); }
}

public void run() {
    String message;
    while(true) {
        try {
            message = userDIS.readUTF();
            if(message.equals(secretCode+"Logout")){
                System.out.println("id"+userID+" Client disconnected.");
            }
            else if(message.length()>30 &&
                    message.substring(0,36).equals(secretCode
                    +"File")) {
                tirth.FILE_TO_RECEIVE = message.substring(36);
                tirth.receiveAndSendFileToAll(userSocket, userID);
            }
            else{
                tirth.sendMessageToAll(message);
                tirth.MessageHistory(message);
            }
        } catch(Exception e){ }
    }
}

public void sendMessage(String s) {
    try {
        userDOS.writeUTF(s);
    } catch(Exception e){ }
}

public void sendFile(String filename) {
    try {
        userDOS.writeUTF(secretCode+"File"+filename);
    } catch(Exception e) { System.out.println("Exception in sendFile() method."); }
}
}

```

6.2 Client.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Scanner;
import java.io.*;
import java.net.*;

public class Client {
    static Scanner in = new Scanner(System.in);
    static DataOutputStream dos;
    static DataInputStream dis;
    static Socket s;
    static boss server;
    public static String username;
    public static int port = 7777;
    public static String secretCode = "46511231dsfdfsdsd#@$#$$#@^$%#@*$#^";
    //-----For File sharing-----
    static FileInputStream fis = null;
    static BufferedInputStream bis = null;
    static OutputStream os = null;
    static InputStream is = null;
    static FileOutputStream fos = null;
    static BufferedOutputStream bos = null;
    static String FILE_TO_RECEIVE;
    static int bytesRead;
    static int current = 0;
    //-----

    static JTextPane chatMessages = new JTextPane();
    static JScrollPane JPchatMessages = new JScrollPane(chatMessages);
    static JLabel helloUser;
    static String msgHistory = new String("");

    public static void main(final String args[]) throws IOException {
        JFrame LoginScreen = new JFrame("Login to LAN Chat & File Sharing");
        JFrame ChatScreen = new JFrame("LAN Chat & File Sharing");
        JFrame FileSharingScreen = new JFrame("Choose a file to send");

        //----- CHAT SCREEN -----
        ChatScreen.setSize(400,650);
        ChatScreen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ChatScreen.setLayout(new GridBagLayout());

        helloUser = new JLabel("Hello and Welcome !");
```

```

ChatScreen.add(helloUser, new GridBagConstraints(0, 0, 1, 1, 3, 1,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,new
    Insets(0,0,0,0),0,0));

JButton logOutButton = new JButton("Logout");
ChatScreen.add(logOutButton, new GridBagConstraints(2, 0, 1, 1, .25, 1,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,new
    Insets(0,0,0,0),0,0));
logOutButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent logOutButtonClick) {
        try {    ChatScreen.setVisible(false);
                System.exit(0);
            } catch(Exception e){ }
        }
    });

chatMessages.setEditable(false);
ChatScreen.add(JPchatMessages, new GridBagConstraints(0, 1, 3, 1, 1.0, 100.0,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,new
    Insets(0,0,0,0),0,0));

JTextField message = new JTextField(20);
ChatScreen.add(message, new GridBagConstraints(0, 2, 1, 1, .5, 1,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,new
    Insets(0,0,0,0),0,0));

JButton send = new JButton("Send");
ChatScreen.add(send, new GridBagConstraints(1, 2, 1, 1, .25, .25,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,new
    Insets(0,0,0,0),0,0));
send.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent sendButtonClick) {
        String msg = message.getText();
        message.setText(null);
        try { if(!msg.equals("")) dos.writeUTF(username + " : " + msg);
            } catch(IOException e){ }
        }
    });

JButton sendFile = new JButton("Upload File");
ChatScreen.add(sendFile, new GridBagConstraints(2,2,1,1,.25,.25,
    GridBagConstraints.CENTER, GridBagConstraints.BOTH,new
    Insets(0,0,0,0),0,0));
sendFile.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent sendFileButtonClick) {
        FileSharingScreen.setVisible(true);
    }
    });

//-----

```

```

//----- LOGIN SCREEN -----
LoginScreen.setLayout(new GridBagLayout());
LoginScreen.setSize(400,650);
LoginScreen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//---For bg image-----
JLabel background=new JLabel(new ImageIcon("Logo.png"),JLabel.CENTER);
LoginScreen.add(background, new
    GridBagConstraints(0,0,1,1,2.0,1.0,GridBagConstraints.CENTER,
        GridBagConstraints.BOTH,new Insets(0,0,0,0),0,0));
//-----

//-----Label-----
JLabel enter = new JLabel("Enter your name");
LoginScreen.add(enter, new
    GridBagConstraints(0,2,1,1,1.0,1.0,GridBagConstraints.CENTER,
        GridBagConstraints.NONE,new Insets(0,0,0,0),0,0));
enter.setFont(new Font(enter.getFont().getName(), Font.BOLD, 16));
//-----

//-----Username Textfield-----
JTextField usernameTextArea = new JTextField(12);
LoginScreen.add(usernameTextArea, new
    GridBagConstraints(0,3,1,1,1.0,1.0,GridBagConstraints.CENTER,
        GridBagConstraints.NONE,new Insets(0,0,0,0),0,0));
usernameTextArea.setFont(new
    Font(usernameTextArea.getFont().getName(),Font.BOLD,16));
//-----

//-----Label-----
JLabel enterPort = new JLabel("Enter Port Number");
LoginScreen.add(enterPort, new
    GridBagConstraints(0,7,1,1,1.0,1.0,GridBagConstraints.CENTER,
        GridBagConstraints.NONE,new Insets(0,0,0,0),0,0));
enterPort.setFont(new Font(enterPort.getFont().getName(), Font.BOLD, 12));
enterPort.setVisible(false);
//-----

//-----PortNumber Textfield-----
JTextField portTextArea = new JTextField(4);
LoginScreen.add(portTextArea, new
    GridBagConstraints(0,8,1,1,1.0,1.0,GridBagConstraints.CENTER,
        GridBagConstraints.NONE,new Insets(0,0,0,0),0,0));
portTextArea.setVisible(false);
//-----

```



```

//-----PrivateChat Option-----
JRadioButton privateChat = new JRadioButton("Private Chat");
LoginScreen.add(privateChat, new
    GridBagConstraints(0,5,1,1,1.0,1.0,GridBagConstraints.CENTER,
        GridBagConstraints.NONE,new Insets(0,0,0,0),0,0));
privateChat.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ChangeEvent) {
        try {
            if(privateChat.isSelected()){
                enterPort.setVisible(true);
                portTextArea.setVisible(true);
            }
            else{
                enterPort.setVisible(false);
                portTextArea.setVisible(false);
            }
        } catch(Exception e) { System.out.println("Server unavailable."); }
    }
});

//-----Login Button-----
JButton login = new JButton("Join chat");
LoginScreen.add(login, new
    GridBagConstraints(0,10,1,1,1.0,1.0,GridBagConstraints.CENTER,
        GridBagConstraints.NONE,new Insets(0,0,0,0),0,0));
login.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent buttonClick) {
        try {
            username = usernameTextArea.getText();
            if(!portTextArea.getText().equals("")) port =
                Integer.parseInt(portTextArea.getText());
            LoginScreen.setVisible(false);

            s = new Socket("192.168.43.198", port);
            dos = new DataOutputStream(s.getOutputStream());
            dis = new DataInputStream(s.getInputStream());
            server = new boss(dis);
            Thread t = new Thread(server);
            t.start();
            System.out.println("Connected to server...");
            ChatScreen.setVisible(true);
        } catch(IOException e) { System.out.println("Server unavailable."); }
    }
});

//-----
LoginScreen.setVisible(true);
//-----

```

```

//----- FILE SELECTOR -----
FileSharingScreen.setSize(500,700);
FileSharingScreen.setLayout(new GridBagLayout());

JFileChooser fileSelector = new JFileChooser();
FileSharingScreen.add(fileSelector, new
    GridBagConstraints(1,0,1,1,.25,.25,GridBagConstraints.CENTER,GridBagConstr
        aints.BOTH,new Insets(0,0,0,0),0,0));
fileSelector.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent fileChooserEvent) {
        String command = fileChooserEvent.getActionCommand();
        if(command.equals(JFileChooser.APPROVE_SELECTION)) {
            File fileToBeSent = fileSelector.getSelectedFile();
            FileSharingScreen.setVisible(false);

            FILE_TO_RECEIVE = fileToBeSent.getName();
            try {
                byte [] fileByteArray = new byte [(int)fileToBeSent.length()];
                dos.writeUTF(username+" : Uploaded
                    \""+FILE_TO_RECEIVE+"\"");
                fis = new FileInputStream(fileToBeSent);
                bis = new BufferedInputStream(fis);
                bis.read(fileByteArray,0,fileByteArray.length);
                os = s.getOutputStream();
                dos.flush();
                os.flush();
                dos.flush();
                dos.writeUTF(secretCode+"File"+FILE_TO_RECEIVE);
                os.write(fileByteArray,0,fileByteArray.length);
                os.flush();
                System.out.println("File Successfully Sent.");
            } catch(Exception e){ }
        }
    }
});

//-----

Runtime.getRuntime().addShutdownHook(new Thread() {
    public void run() {
        try {
            dos.writeUTF("> "+username+" logged out...");
            dos.writeUTF(secretCode+"Logout");
            System.out.println("Logged out....");
        } catch(Exception e) { }
    }
});
}

```

```

public static void setUsername(String str) {
    if(username.equals("")) username = str;
    helloUser.setText("Hello "+username+". Welcome!");
    try {
        dos.writeUTF("> "+username+" logged in...");
    } catch(IOException e){ }
}

public static void updateMessageArea(String msg) {
    msgHistory = msgHistory + msg + "\n";
    chatMessages.setText(msgHistory);
}

public static void reconnect() {
    try {
        s.close();
        s = new Socket("192.168.43.198", port);
        dos = new DataOutputStream(s.getOutputStream());
        dis = new DataInputStream(s.getInputStream());
        server = new boss(dis);
        Thread newConnection = new Thread(server);
        newConnection.start();
    }
    catch(Exception e) {
        System.out.println("Exception caught in reconnect().");
    }
}

public static void receiveFile() {
    String workingDir = System.getProperty("user.dir");
    String FILE_TO_RECEIVED = workingDir + "\\Files\\" + FILE_TO_RECEIVE;
    File myFile = new File(FILE_TO_RECEIVED);
    try {
        current = (int)myFile.length();
        System.out.println("File " + FILE_TO_RECEIVED + " downloaded (" + current +
            " bytes read)");
    }
    catch(Exception e) {
        System.out.println("Exception caught in receiveFile().");
    }
}
}

```

```

class boss extends Thread {
    DataInputStream disServer;
    public boss(DataInputStream z) {
        disServer = z;
    }

    public void run() {
        while(true) {
            try {
                String str = disServer.readUTF();
                if(str.length()>30 && str.substring(0,36).equals(Client.secretCode+"File")) {
                    Client.FILE_TO_RECEIVE = str.substring(36);
                    Client.receiveFile();
                }
                else if (str.length()>30 && str.substring(0,36).equals(Client.secretCode+"User"))
                    Client.setUsername(str.substring(32));
                else
                    Client.updateMessageArea(str);
            } catch(Exception e) {
                System.out.println("Exception in run method. Reconnecting....");
                Client.reconnect();
                break;
            }
        }
    }
}

```

Logo.png

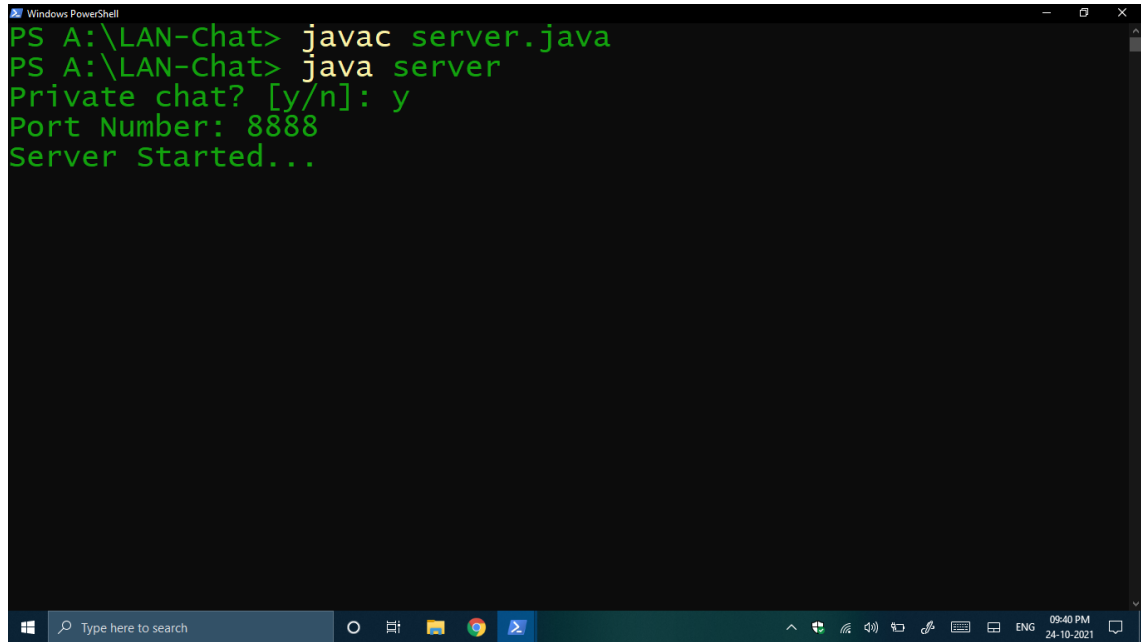


7. OUTPUT SCREENS

7.1 Running Server & Client

- **Running Server**

To run the application first we need to run the server

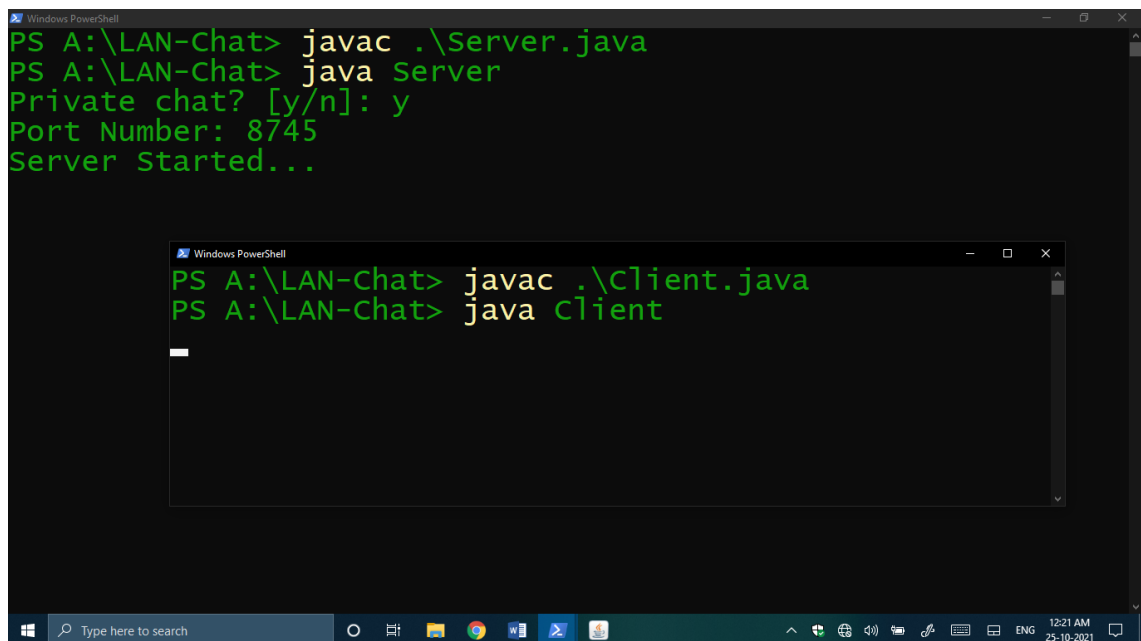


```
Windows PowerShell
PS A:\LAN-Chat> javac server.java
PS A:\LAN-Chat> java server
Private chat? [y/n]: y
Port Number: 8888
Server Started...
```

Fig-7.1: Server

- **Running Client**

After running the server we can run Client & connect to the server



```
Windows PowerShell
PS A:\LAN-Chat> javac .\Server.java
PS A:\LAN-Chat> java Server
Private chat? [y/n]: y
Port Number: 8745
Server Started...
```

```
Windows PowerShell
PS A:\LAN-Chat> javac .\Client.java
PS A:\LAN-Chat> java client
```

Fig-7.2: Client

7.2 Login Screen

When we run the Client the Login Screen will appear as follows. You can enter your name and login or click on the “Private Chat” radio button & enter the port number for private chat.

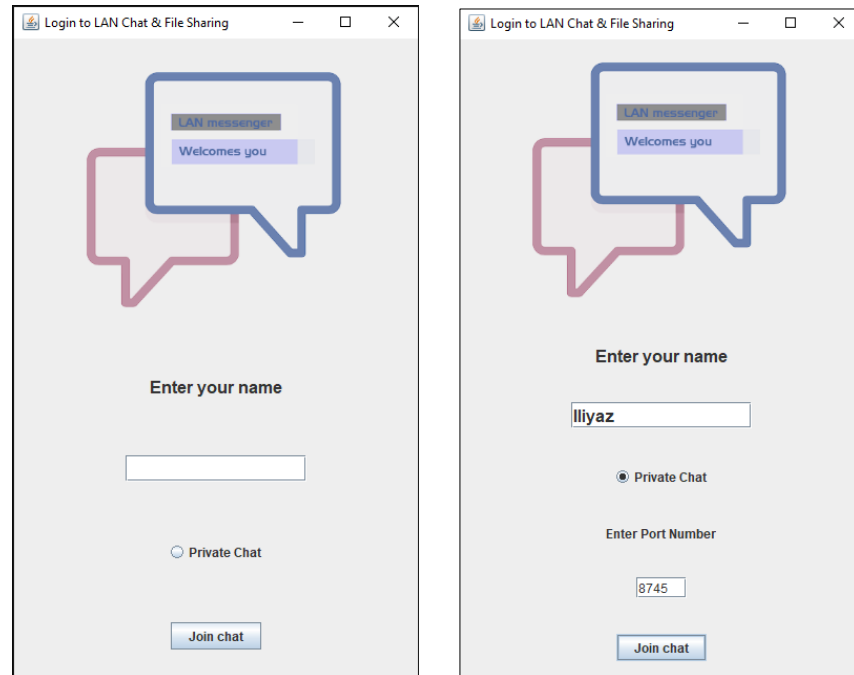


Fig-7.3: Login Screen

- **Connecting multiple clients**

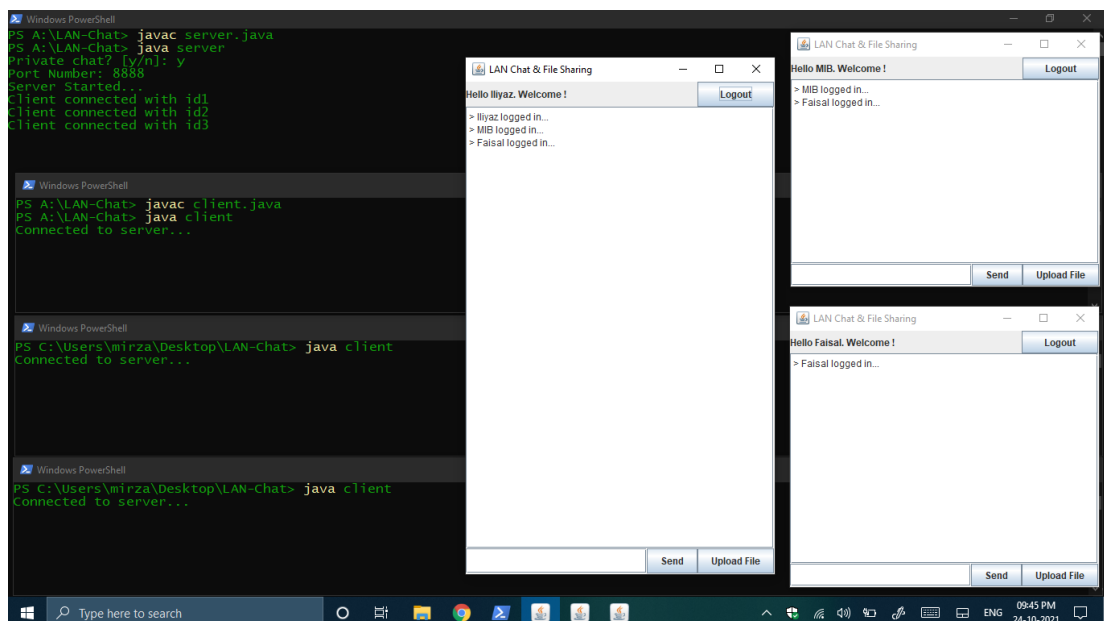


Fig-7.4: Multiple Clients

7.3 Chat Screen

After successful login chat screen will appear as follows and now you can chat with other connected clients

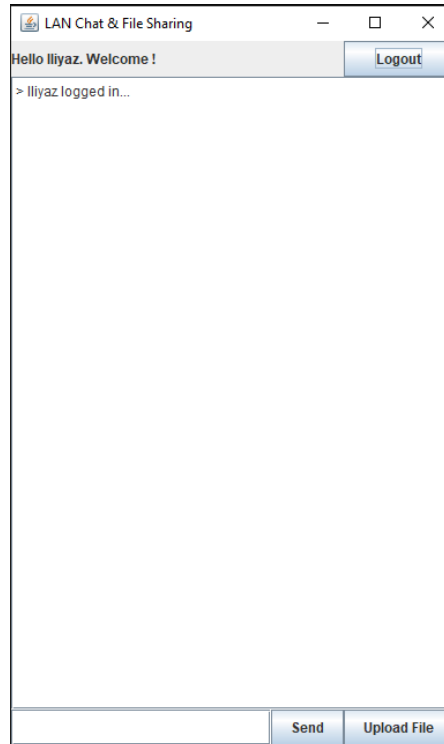


Fig-7.5: Chat Screen

- Chatting with multiple clients

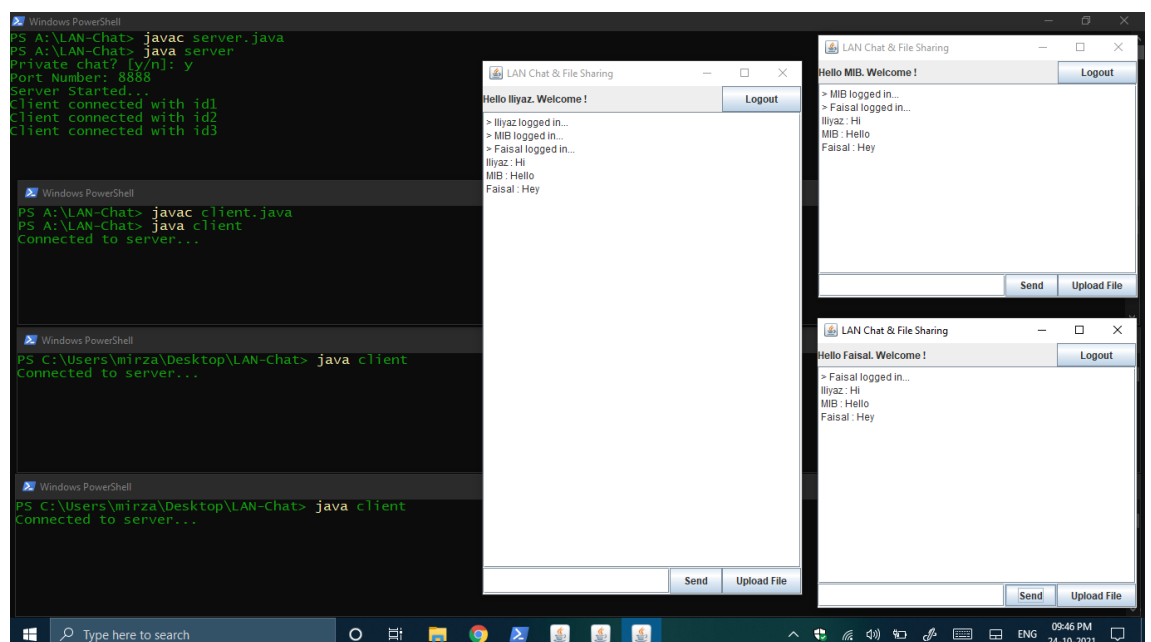


Fig-7.5: Multiple Clients Chatting

7.4 File Selector Screen

To share the file there is a button “Upload File”, if you want to share any file you can click on it, which will open file selector screen. Select the file which you want to send & click open

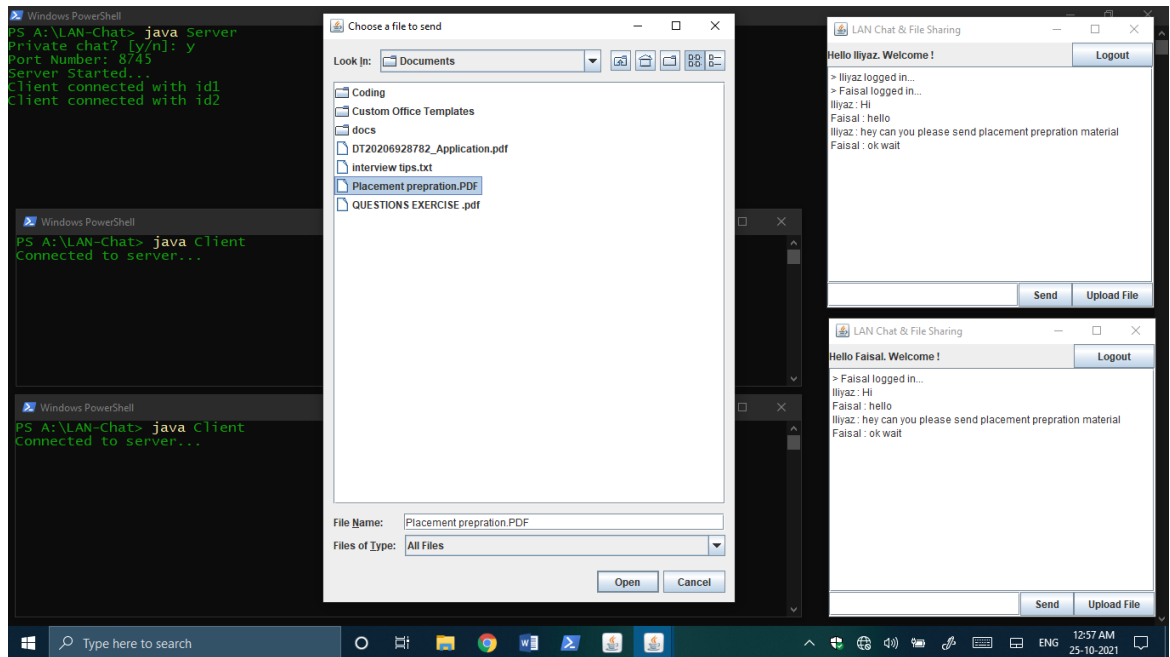


Fig-7.6: File Selector Screen

Then the selected file be sent and all clients gets an uploaded message and the path where it is stored.

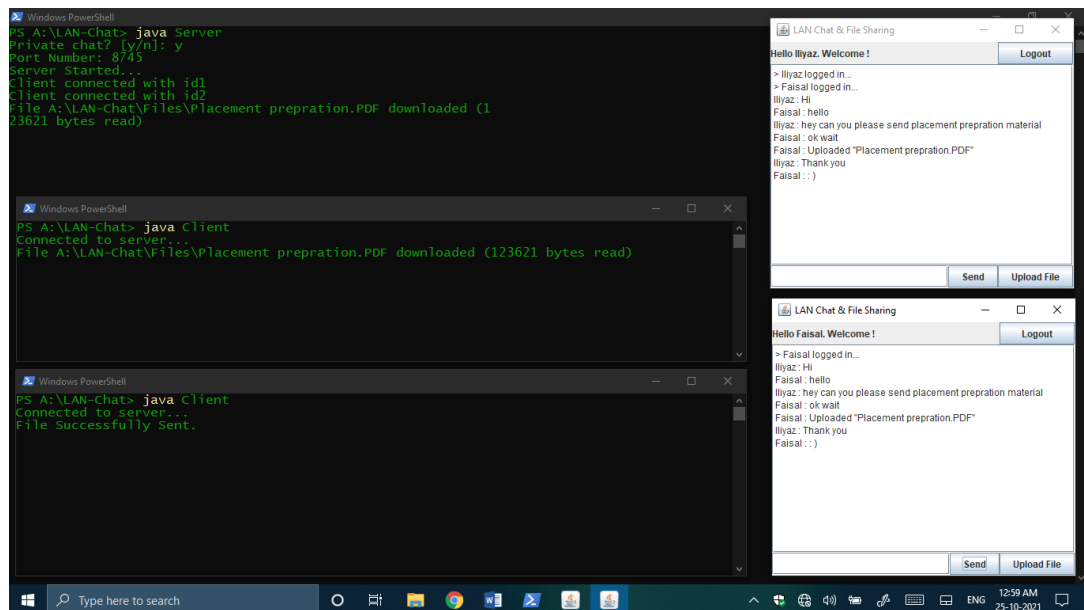


Fig-7.7: File Uploaded

So, whoever needs that file, they can access/open it from the specified path for receiving file as shown below.

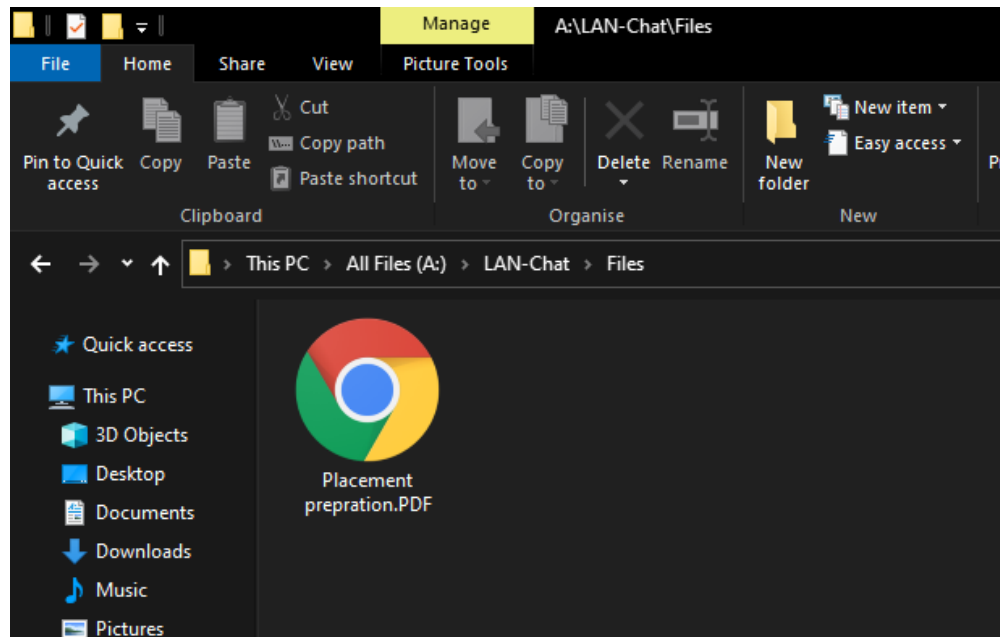


Fig-7.8: Uploaded File Path

7.5 Logout

To logout/exit the application there is a “Logout” button on the top right corner. Clicking on which will log you out.

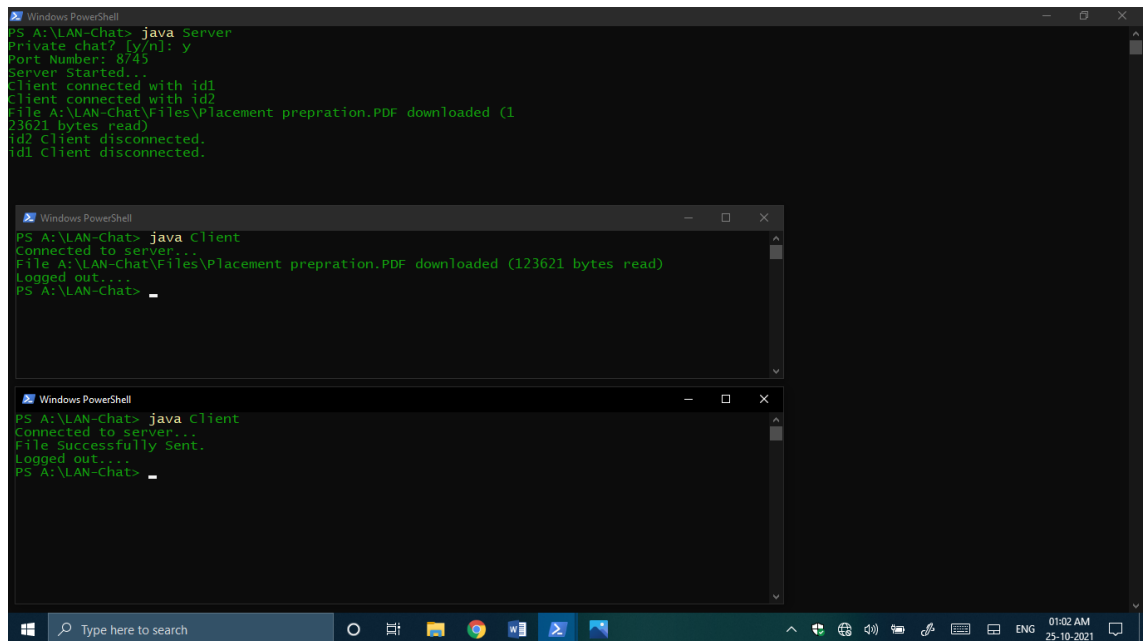


Fig-7.9: Logout

7.6 Chat History

This application also maintains a chat history, whenever there is a need of chat history the user can access it from the specified path for Chat History as shown below

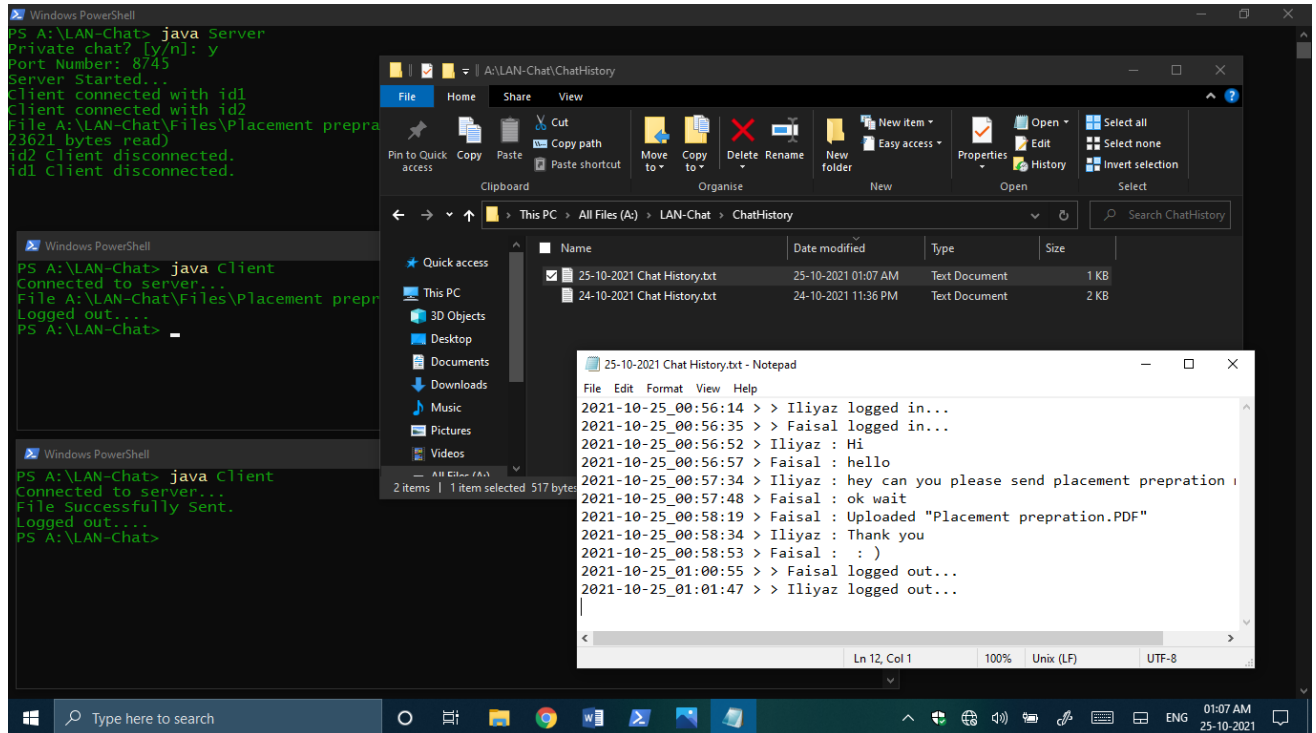


Fig-7.10: Chat History

8. CONCLUSIONS AND DISCUSSION

The LAN-Chat is a solution to some communication problems and reduces the use of resources including time factor of an organization's internal communication. The proposed system enables users to communicate on networks outside internet boundaries. it can be integrated in other application areas such as in school, university and public libraries for communication between library patrons and librarian, in business offices, scientific organizations, and the academe, to mention a few. The proposed system can be a future replacement for many internet chat applications and will cost the organization lesser resources to implement. The strengths of the system identified are: more communication opportunities within an organization without using an internet subscription; it can work on all Windows platforms provided that the client and server are using the same type of operating system; it enables connected clients to chat with each other & upload files; it supports peer-to-peer and server-based modes; user interface is common and familiar; and users may create chat rooms for specific topics or users so if there is any urgent management announcement is easy to broadcast to everyone at once and to specific persons only. There are other advantages of the system especially when future enhancements would be analysed and implemented.

9. REFERENCES

- [1] <https://www.javatpoint.com/socket-programming>
- [2] <https://www.javatpoint.com/java-swing>
- [3] https://www.youtube.com/watch?v=ulHkiqGXrFU&ab_channel=TeachMeIDEA
- [4] <https://www.tutorialspoint.com/swing/index.htm>
- [5] <https://www.geeksforgeeks.org/socket-programming-in-java/>
- [6] <https://docs.oracle.com/javase/tutorial/networking/sockets/>
- [7] <https://mail.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket>
- [8] <https://www.c-sharpcorner.com/article/how-to-make-a-chat-application-using-sockets-in-java/>
- [9] <https://www.geeksforgeeks.org/multi-threaded-chat-application-set-1/>
- [10] <https://medium.com/nerd-for-tech/create-a-chat-app-with-java-sockets-8449fdaa933>