

ONLINE TEXT EDITOR USING SOCKET PROGRAMMING

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

CSE302: COMPUTER NETWORKS

Submitted by

SHUBHANKIT CHOWDHURY

(Reg.no: 124003295, B. Tech CSE)

December 2022



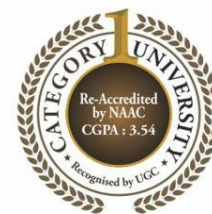
SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 of the UGC Act, 1956)



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401



THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING THANJAVUR -613 401

Bonafide Certificate

This is to certify that the report titled “**Online Text Editor Using Socket Programming**” submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Shri SHUBHANKIT CHOWDHURY (Reg. No.124003295,CSE)** during the academic year 2022-23, in the School of Computing .

Project Based Work *Viva voce* held on _____

Examiner 1

Examiner 2

LIST OF FIGURES

Figure No.	Title	Page No.
1	Client server architecture	1
2	File Transfer Protocol	3
3	Connection through ports	5
4	Working of client and server	6
6-14	Snapshots of output and execution	30-37

LIST OF TABLES

Table No.	Title	Page No.
1.1.6	Socket methods	6

ABBREVIATIONS

TCP	Transfer Control Program
IP	Internet Protocol
UDP	User Datagram Protocol
GUI	Graphical User Interface
FTP	File Transfer Protocol

ABSTRACT

This project implements the working of online text editor.

Sockets provide the communication between two computer using transmission control protocol. One is client and another is server. A client program creates a socket and connects with the server socket. If the connection is success the server creates one socket object. Now the server and client can make the communication. This project is implemented by using java programming language. This chat application makes use of java's GUI swing. The messages are displayed in the GUI window which has similar fuctionalities like Google Docs. One of the advantages of using java language for networking is it was initially designed with networking in mind. It provides security. Network programs can be easily implemented in java. It includes classes for network programs to communicate with types of servers.

KEY WORDS: client and server, socket programming, TCP, java networking

TABLE OF CONTENTS

Title	PageNo
Bonafide certificate	ii
List of Figures	iii
List of tables	iv
Abbreviations	v
Abstract	vi
Chapter 1 Introduction, Merits and Demerits	1
1.1 Proposed work	1
1.2 Communication protocols	1
1.3 Sockets	1
1.4 File Transfer Protocol	2
1.5 Types of Socket	3
1.6 Software used	4
1.7 Working	4
1.8 Merits	7
1.9 Demerits	7
Chapter 2 Source code	8
Chapter 3 Snapshots	26
Chapter 4 Conclusion and future plans	33
Chapter 5 References	34

CHAPTER 1

INTRODUCTION, MERITS AND DEMERITS

1.1 PROPOSED WORK

This project discusses the detailed overview of developing a online text editor using socket programming. I have developed a client server based text editor using socket programming in Java programming language.

1.2 COMMUNICATION PROTOCOLS

There are two types of network communication. The Transfer control protocol and User datagram protocol. The TCP (stream communication) provides a connection oriented service. It enables a client to make communication with the server and the client and the server can start to communicate. It provides a sequenced flow of error free data. The UDP (datagram communication) is a connectionless protocol. Not all the packets are required to arrive at the destination.

1.3 SOCKETS

A Socket is an end point of communication between 2 devices which is bound to a port number so that the TCP can identify the application to which the data is being sent. A port number is used to identify your application on a network. The sockets make use of TCP to provide communication.

In this chat application, there is a client application and a server application. Depending on the port number, a device may have “n” number of sockets. Different types of protocol have different port number. Socket programs are mostly used to communicate between various processes running on different systems. It is generally used to create a client-server environment. The client is someone who requests. The server is someone who responds.

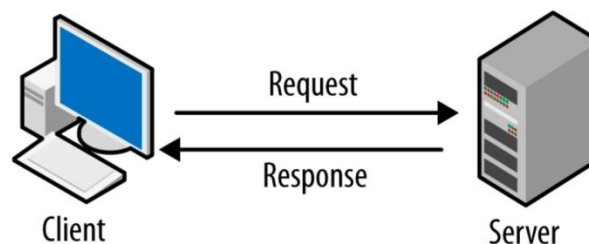


FIG 1: CLIENT SERVER ARCHITECTURE

For a computer to send and receive message they need to have sockets at both the ends. Initially only the client will have the socket. The server application will have a server socket object at its side. The job of server socket object is to wait for the client's request. The server socket object takes an argument: port number. The socket object takes two arguments: IP address and port number. The IP address or Internet Protocol address is the unique address used to identify our computer on the network. Now once the client makes a request a socket object is created at server side. Since both the socket objects are present, a connection is established between the client and the server.

1.4 FILE TRANSFER PROTOCOL:

One of the common Internet protocols offered by TCP/IP is FTP, which is used to transfer files from one host to another, usually between a client and a server. File transfer protocol, or FTP, is a method commonly used to transfer files over a network. Compared to any other alternative ways, it supports great speed. Therefore, it is frequently utilised whenever someone has to upload or download from a web host.

When an FTP session is initiated between a client and a server, the client opens a control TCP connection with the server-side. Over this, the client transmits control information. In response, the server opens a data connection to the client-side. Over a single data connection, just one file may be transmitted. But throughout the user session, the control connection is still active. Since HTTP is stateless, it is not required to maintain any user state. However, FTP must continue to track its user's state during the session.

FTP's capacity to handle huge file transfers is one of the primary reasons why contemporary enterprises and people require it. Most techniques work when transferring a relatively modest file, like a Word document, however FTP allows you to send hundreds of gigabytes at once while maintaining a seamless transmission.

Having the capacity to send more data also enhances workflow. You can choose a few files and transmit them all at once using FTP because it lets you send numerous files at once. Without FTP, you might have to transmit them one at a time when you could be working on anything else.

You can use FTP to send them all at once, for instance, if you need to transfer a sizable number of crucial documents from headquarters to a satellite office but need to be at a meeting in six minutes. FTP can handle the transfer even if it takes 15 minutes to finish, freeing you up to attend meeting.

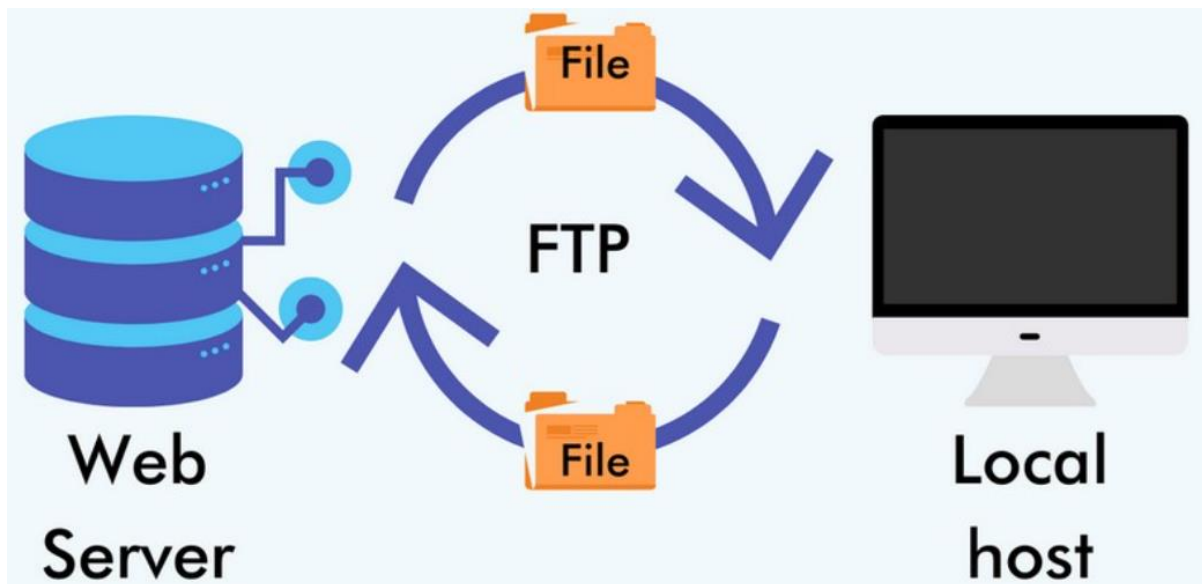


FIG 2: FTP

Types of FTP servers:

Anonymous Server: It is a common FTP server, that is for all clients. No password is required to access this server. Most clients use it.

Non-Anonymous Server: The non-Anonymous server is a paid server. If the user uses the non-anonymous server, the user needs the relevant password to access the file or to enter into the server.

1.5 TYPES OF SOCKET:

The sockets are divided into three different types.

- **Stream sockets:** A stream socket allows communication using transfer control protocol. Stream sockets provide reliable, bidirectional, ordered and unimitated flow of data. The type of socket used is SOCK_STREAM. The data can be read and written to these sockets as a byte stream.
- **Datagram sockets:** A datagram socket allows communication using user datagram protocol. It supports bidirectional flow of messages. The delivery is not guaranteed. They provide connectionless communication. The receiving data would not be in the same order as sent. The type of socket used is SOCK_DGRAM.
- **Raw sockets:** Raw sockets give access to Internet Message Control Protocol (ICMP). They are normally datagram oriented. They are not intended for general users. Most applications don't use raw sockets. The type of socket used is SOCK_RAW The type of socket used in this chat application is stream sockets.

1.6 SOFTWARE USED

This application is developed using Java programming language on windows operating system. Java was the first programming language which was designed having network foremost in the mind. Java's GUI swing concept and event listeners are used in the designing of this application. The chat application is designed like Google Docs. It gives the users the experience of real time text manipulation.

Network programming using java have advantages over any other programming language.

- Java library includes classes for communicating using TCP/IP protocols. We have different socket class for both the client and the server.
- Java provides solutions to problems such as security, platform independent. It also supports code portability.
- It makes writing network programs easier than writing network programs in any other languages. They are flexible.
- Java sockets if efficiently used can cause low network traffic.
- Java sockets hides much details in existing socket programming.
- The Java applets can also communicate across the internet. These features allow programmers to search the internet without worrying that the program may crash the systems, steal the data or could spread the virus.
- It has Secure Socket Layer (SSL) which has provides more security, message privacy, authentication and integrity benefits.

1.7 WORKING

This application implements communication between a single client and a single server. It has two java class files namely: myserver.java and myclient.java. The java.net.Socket class indicates a socket and the java.net.ServerSocket is used to listen and make connection with the clients. First we start the server application so that the socket server objects starts waiting for client's request.

The following steps occur when establishing the TCP connection between the client and server:

- The server starts the ServerSocket object and it indicates which port number the communication is going to occur on. Figure 2 demonstrates the connection to sockets through a common port number.

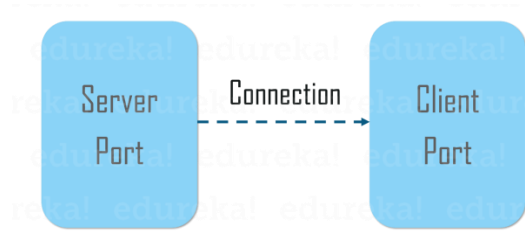


FIG 3: CONNECTION THROUGH PORTS

- The server invokes the `accept()` method. It waits till the client makes connection to the given port number.
- The client then initiates the socket object with the same server name and the port number to connect to.
- If the communication is successful, the client has a socket object which can communicate with the server. On the server side the `accept()` method returns a reference to a new socket on the server that is connected to the client's server
- Once the communication is over one or both the sides close the connection

Figure 3 demonstrates the working of client and server. Once the communication is successful they can communicate using the I/O streams: `InputStream` and `OutputStream`. The client's `InputStream` is connected to server `OutputStream` and the server `OutputStream` is connected to client's `InputStream`. The code handles exception, AWT were also used in developing the application.

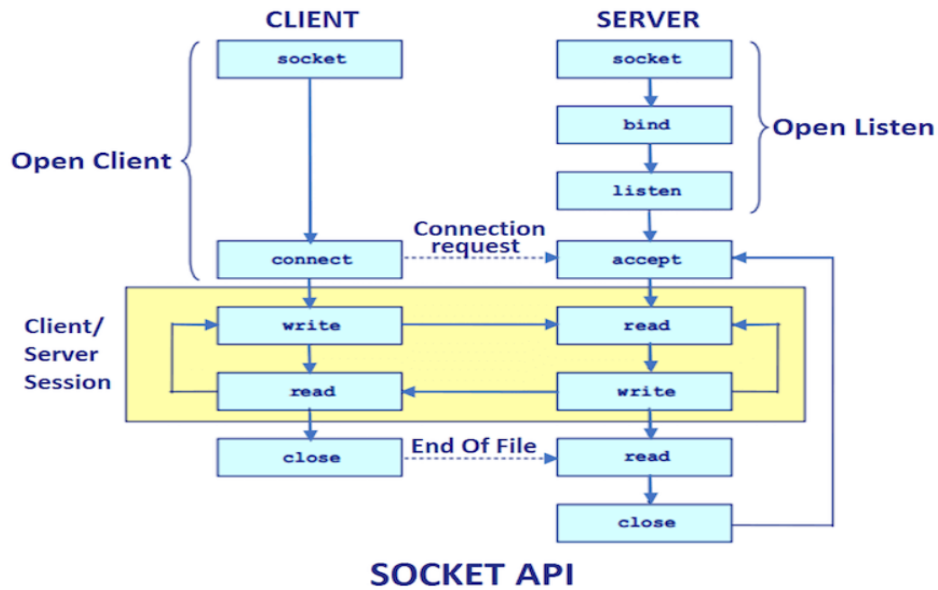


FIG 4: WORKING OF CLIENT AND SERVER

Socket()	A new socket is created at its end
Bind()	Binds the socket to port and local IP address
Listen()	Used in passive open
Connect()	Starts connection to the other socket
Accept()	Accepts new connection from the other sockets
getInputStream()	It returns the input stream that is attached with this socket
getOutputStream()	It returns the output stream that is attached with this socket
Close()	Closes the socket()

TABLE 1.1.6 SOCKET METHODS

1.8 MERITS

- The information can be encrypted
- Sockets causes low network traffic
- Since the data transmission is byte level, it can be customized.
- The data security is strong.
- The client and server can exchange information in real time.
- The performance is high and the data transmission time is short.
- One of the advantages of using sockets is its transparency. It means even the program was written in java or other programming languages it will still be able to communicate with other languages such as Python or C++.
- Speed is one of the advantages of File Transfer Protocol.
- Efficiency is more in FTP.
- File sharing is also an advantage of FTP which takes place between two machines on a network

1.9 DEMERITS

- Only raw data can be transferred between applications. Both the server and client have to give some means to make the data helpful in some way.
- Because of the security, the java applets running in a web browser can only provide connections only with the machine requested and nowhere else on the network.
- File size limitations are an issue with FTP. File transfer limit is upto 2 GB.
- The FTP does not allow multiple receivers.
- FTP is unprotected. Although we utilise login IDs and passwords to make it secure, hackers can still target them.

CHAPTER 2

SOURCE CODE

There are two programs “Server.java” and “Client.java”

Server.java

```
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Vector;

public class Server {

    static Vector<ClientHandler> ClientsVec = new Vector<>();

    static int nClients = 0; //numbers of clients connected to server

    @SuppressWarnings("resource")
    public static void main(String[] args) throws IOException {

        ServerSocket server = new ServerSocket(9999);

        System.out.println(" Server is waiting for the client ");
```

```

// running infinite loop for getting
// client request
while (true) {

    Socket SocServer = server.accept();

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

    System.out.println("The client with port number = " + SocServer.getPort() + "
Added to the server");

    // obtaining input and out streams

    System.out.println("waiting for the ID of the Client....");

    ObjectInputStream ServerInput ;

    ServerInput= new ObjectInputStream(SocServer.getInputStream());

    ObjectOutputStream ServerOutput ;

    ServerOutput = new ObjectOutputStream(SocServer.getOutputStream());

    String id = ServerInput.readUTF();

    System.out.println("Client ID : " + id);

    ClientHandler ClientsHand ;

    ClientsHand= new ClientHandler(SocServer, id, ServerInput, ServerOutput);

    //to handle multiple clients connected to server

    // Creatio of a new Thread with this object.

```



```

        Thread thread = new Thread(ClientsHand); //for each client one thread

        System.out.println(" Client is being added to active client list...");

        // add this client to active clients list
        ClientsVec.add(ClientsHand);

        // start the thread.
        thread.start();

        nClients++;

    }
}

class ClientHandler implements Runnable { //Runnable is an interface that is to be implemented by a class whose instances are intended to be executed by a thread

    static Vector<ClientHandler> NEWClientsVec = new Vector<>();

    final ObjectInputStream input;

    final ObjectOutputStream output;

    DataOutputStream dout;

    public String ID;

    Socket s;

```

```

// Constructor

public ClientHandler(Socket s, String ID, ObjectInputStream input, ObjectOutputStream
output) {

    this.ID = ID;

    this.s = s;

    this.input = input;

    this.output = output;


    try {
dout = new DataOutputStream(s.getOutputStream());
    } catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
    }
}

public void run() {

    try
    {
        while (true) {

            System.out.println("This text is send by client to be shared...");


            // receive "share"

            String received = input.readUTF();

```

```

if (received.equals("Share")) {

    // receive the IDs

    received = input.readUTF();

    NEWClientsVec.clear(); //the clients to which data is intended to share

    String[] IDs = received.split("\\-"); //suppose the ids passed have // then demar-
    cation is done based on this

    for (String s : IDs) {

        for (ClientHandler mc : Server.ClientsVec) { //ClientsVec is all the active cli-
        ents connected to server including to sender

            if (s.equals(mc.ID) && !mc.ID.equals(this.ID)) { //sender cannot send it to
            itself so, that and condition

                NEWClientsVec.add(mc);

                break;

            }

        }

        // when the the other clients want to edit show this at the parent

        NEWClientsVec.add(this);

    }

}

else if(received.equals("Upload"))

{

    try{

        String rcv = input.readUTF();

        System.out.println(rcv);

```

```

File fi = new File("E:\\server\\" + rcv);

rcv = input.readUTF();

System.out.print(rcv);// Create a file writer that doesn't append

FileWriter wr = new FileWriter(fi, false); //false so, that if u select a file that al-
ready has text, new text will overwrite previous text

//if true, new text will be appended to old tex

// Create buffered writer to write

BufferedWriter w = new BufferedWriter(wr);


// Write

w.write(rcv);

w.flush();

w.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}

else{

received = input.readUTF(); //taking the input text

for (ClientHandler mc : NEWClientsVec){

if (!ID.equals(mc.ID)) { //sender cannot send text to itself, so that if condition

```

```
        mc.output.writeUTF(received);

        mc.output.flush(); //clear the buffer after every operation
    }

}

}

}

}

}

}

}

}

}
```

Client.java

```
import javax.swing.*;
import java.awt.*;
import javax.swing.plaf.metal.MetalLookAndFeel;
import javax.swing.plaf.metal.OceanTheme;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.*;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

class Client extends JFrame implements ActionListener, KeyListener {
    public static Socket socClient;

    public static ObjectInputStream ClientInput;
    public static ObjectOutputStream ClientOutput;
    public static DataInputStream din;
    public String SelectedText;
    public String ClientIDToShare;

    // Text component
    JTextArea t;

    // Frame
    JFrame f;

    // Main class
    public static void main(String[] args) {
        try {
```

```

        socClient = new Socket("localhost", 9999);
System.out.println("Connected!");
        Client c1 = new Client();
        Scanner scn = new Scanner(System.in);
        ClientOutput = new ObjectOutputStream(socClient.getOutputStream());
        ClientInput = new ObjectInputStream(socClient.getInputStream());
        din = new DataInputStream(socClient.getInputStream());
        System.out.print("Write your ID : ");
        String id = scn.nextLine();
        ClientOutput.writeUTF(id); //output stream writes ID in itself
        ClientOutput.flush();
        System.out.println("Write the name for your frame");
        String filename = scn.nextLine();
        c1.ClientGUI(filename); //attaching the framename to frame top
        System.out.print("Now You Start your Real Connection");
        while (true) {
            String NewDataInTextArea = ClientInput.readUTF();
            //input given by user everytime continuously so while loop
            c1.ChangeText(NewDataInTextArea); //display it on text area
            //server to receiver, to display text on receiver text area

        }
    } catch (UnknownHostException e) {
        e.printStackTrace();
        e.getMessage();
    } catch (IOException e) {

        e.printStackTrace(); //the exceptions occuring on every stage is displayed
    }
}

```

```

        e.getMessage();
    }
}

public void ClientGUI(String str) {
    // Create a frame
    f = new JFrame(str); //Jframe is created with given name
    try {
        // Set look appearance of window and feel behaviour
        UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
        // Set theme to ocean
        MetalLookAndFeel.setCurrentTheme(new OceanTheme());
    } catch (Exception e) {
        e.printStackTrace();
    }

    // Text component
    t = new JTextArea();
    t.setLineWrap(true);


    // Create a menu bar
    JMenuBar mb = new JMenuBar();

    // Create a menu for menu
    JMenu m1 = new JMenu("File");

    // Create menu items
    JMenuItem mi1 = new JMenuItem("New");
    JMenuItem mi2 = new JMenuItem("Open");
    JMenuItem mi3 = new JMenuItem("Save");

```



```

JMenuItem mi10 = new JMenuItem("Share");
JMenuItem upmenuui = new JMenuItem("Upload");
// Add action listener
mi1.addActionListener(this);
mi2.addActionListener(this);
mi3.addActionListener(this);
mi10.addActionListener(this);
upmenuui.addActionListener(this);
//downmenuui.addActionListener(this);
m1.add(mi1);

m1.add(mi2);
m1.add(mi3);
m1.add(mi10);
m1.add(upmenuui);
// Create a menu for menu
JMenu m2 = new JMenu("Edit");
// Create menu items
JMenuItem mi4 = new JMenuItem("cut");
JMenuItem mi5 = new JMenuItem("copy");
JMenuItem mi6 = new JMenuItem("paste");
// Add action listener
mi4.addActionListener(this);
mi5.addActionListener(this);
mi6.addActionListener(this);
m2.add(mi4);
m2.add(mi5);
m2.add(mi6);

```

```

JMenuItem mc = new JMenuItem("close");
mc.addActionListener(this);
mb.add(m1);
mb.add(m2);
mb.add(mc);
f.setJMenuBar(mb);
f.add(t);
f.setSize(500, 500);
f.show();
t.addKeyListener(this);
}

public void ChangeText(String str) {
    t.setText(str);
}

// If a button is pressed
public void actionPerformed(ActionEvent e) {
    String s = e.getActionCommand();
    if (s.equals("cut")) {
        t.cut(); //inbuilt function
    } else if (s.equals("copy")) {
        t.copy();
    } else if (s.equals("paste")) {
        t.paste();
    }
    else if (s.equals("Save")) {

JFileChooser j = new JFileChooser("c:"); //shows default drive save location as c

```

```

int r;

r = j.showSaveDialog(null);

if (r == JFileChooser.APPROVE_OPTION) { //if u select file location where u want to store

    // The label is set to the path of selected directory

    File fi = new File(j.getSelectedFile().getAbsolutePath());

    try {

        // Create a file writer that doesn't append

        FileWriter wr = new FileWriter(fi, false);

        //if true, new text will be appended to old text


        // Create buffered writer to write

        BufferedWriter w = new BufferedWriter(wr); //writing the content in desired location

        // Write

        w.write(t.getText());

        w.flush();

        w.close();

    } catch (Exception evt) {

        JOptionPane.showMessageDialog(f, evt.getMessage());

    }

}

else

    JOptionPane.showMessageDialog(f, "user cancelled the operation");//Popup

}

else if (s.equals("Open")) {

    // Create an object of JFileChooser class

    JFileChooser j = new JFileChooser("f:"); //default location will be f drive

```

```

int r;

r = j.showOpenDialog(null);

// to select the file
if (r == JFileChooser.APPROVE_OPTION) {
    // Set the label to the path of the selected directory
    File fi = new File(j.getSelectedFile().getAbsolutePath());
    try {
        // String
        String s2;
        // File reader
        FileReader fr = new FileReader(fi);
        // Buffered reader
        BufferedReader br = new BufferedReader(fr); //to display a saved file
        // Initailise sl
        String sl;
        sl = br.readLine();
        // Take the input from the file
        while ((s2 = br.readLine()) != null) {
            sl = sl + "\n" + s2;
        }
        // Set the text

        t.setText(sl); //the fetched text from the opened file is displayed in text box
    } catch (Exception evt) {
        JOptionPane.showMessageDialog(f, evt.getMessage());
    }
}

```

```

// to cancel the operation

else

JOptionPane.showMessageDialog(f, "the user cancelled the operation");

}

else if (s.equals("New")) {

t.setText("");

}

else if (s.equals("close")) {

f.setVisible(false); //frame vanishes

try {

ClientInput.close();

socClient.close();

ClientOutput.close();

} catch (IOException e1) {

e1.printStackTrace();

}

} else if (s.equals("Share")) {

try {

ClientOutput.writeUTF("Share"); //sender to server

ClientOutput.flush();

} catch (IOException e1) {

e1.printStackTrace();

}

ClientIDToShare = JOptionPane.showInputDialog("Enter the ID's of the Client to send this text..");

try {

ClientOutput.writeUTF(ClientIDToShare); //tells server that this id will get shared contents

ClientOutput.flush();

```

```

    } catch (IOException e1) {
    e1.printStackTrace(); //exception s stack trace
    }
    }
    else if(s.equals("Upload"))
    {
    JFileChooser j = new JFileChooser("f:"); //default location will be f drive
    // Invoke the showsOpenDialog function to show the save dialog
    int r;
    r = j.showOpenDialog(null);
    // when the user selects a file
    if (r == JFileChooser.APPROVE_OPTION) {
    // Set the label to the path of the selected directory

    File fi = new File(j.getSelectedFile().getAbsolutePath());
    try {
    ClientOutput.writeUTF("Upload");
    ClientOutput.flush();
    ClientOutput.writeUTF(fi.getName());
    ClientOutput.flush();
    } catch (IOException e2) {
    // TODO Auto-generated catch block
    e2.printStackTrace();
    }
    try {
    // String
    String s2;
    // File reader

```

```

FileReader fr = new FileReader(fi);
// Buffered reader
BufferedReader br = new BufferedReader(fr); //to display a saved file
// Initailise sl
String sl;
sl = br.readLine();

// Take the input from the file
while ((s2 = br.readLine()) != null) {
sl = sl + "\n" + s2;
}
// Set the text
//t.setText(sl);
//the fetched text from the opened file is displayed in text box
//SelectedText = t.getText();
try {

ClientOutput.writeUTF(sl); //to server (as u sharing text )
ClientOutput.flush();

} catch (IOException e1) {
e1.printStackTrace();
}

} catch (Exception evt) {
JOptionPane.showMessageDialog(f, evt.getMessage());
}

```

```

    }

}

// If the user cancelled the operation
else
OptionPane.showMessageDialog(f, "the user cancelled the operation");
}

public void keyTyped(KeyEvent e) {}
public void keyPressed(KeyEvent e) {}
public void keyReleased(KeyEvent e) {
    SelectedText = t.getText();
    try {

        ClientOutput.writeUTF(SelectedText); //to server (as u sharing text )
        ClientOutput.flush();
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}

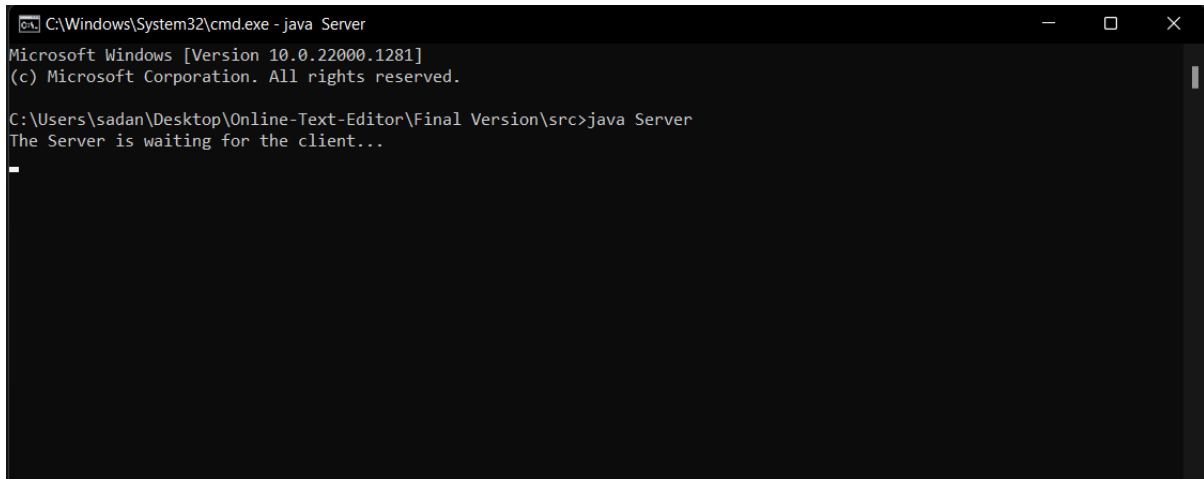
}

```


CHAPTER 3

SNAPSHOTS

When the server is started “The Server is waiting for the client...” message is printed in command prompt

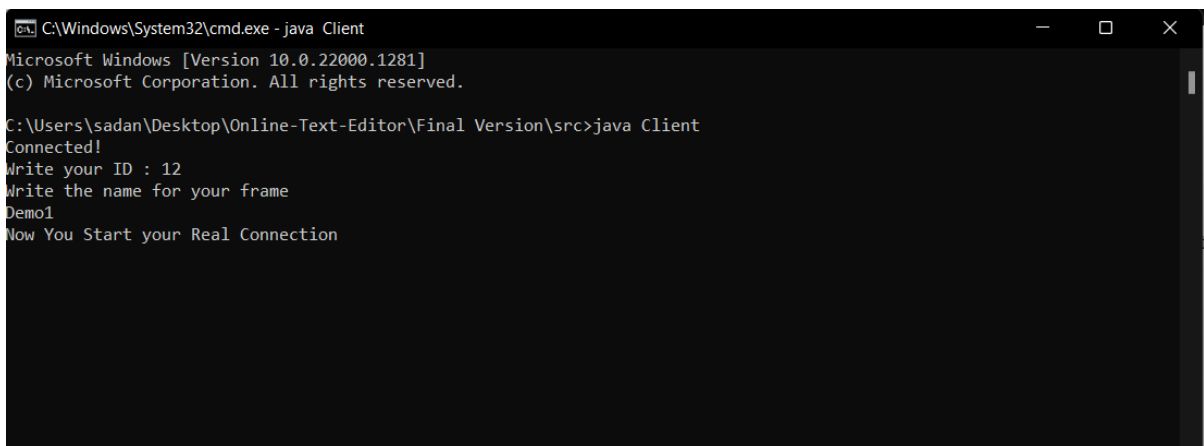


```
C:\Windows\System32\cmd.exe - java Server
Microsoft Windows [Version 10.0.22000.1281]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sadan\Desktop\Online-Text-Editor\Final Version\src>java Server
The Server is waiting for the client...
```

FIG 5

Now, client program is started which asks for “Write your ID : “ and “Write the name for your frame” after giving the input “Now You Start your Real Connection” is displayed on command prompt.



```
C:\Windows\System32\cmd.exe - java Client
Microsoft Windows [Version 10.0.22000.1281]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sadan\Desktop\Online-Text-Editor\Final Version\src>java Client
Connected!
Write your ID : 12
Write the name for your frame
Demo1
Now You Start your Real Connection
```

FIG 6

Now,the client window opens up with the given credentials

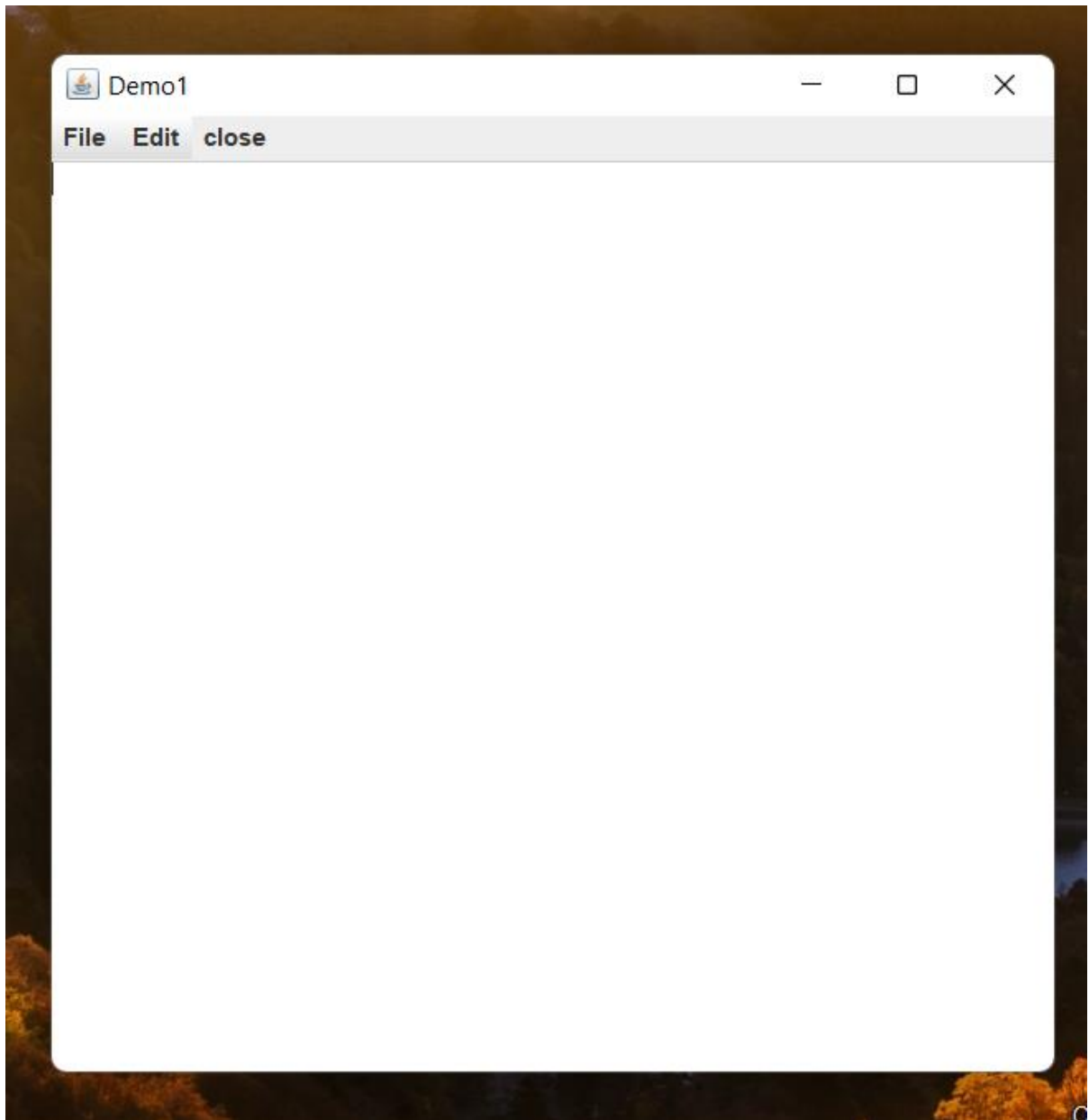


FIG 7

Similarly, we can open another client for communication between them.

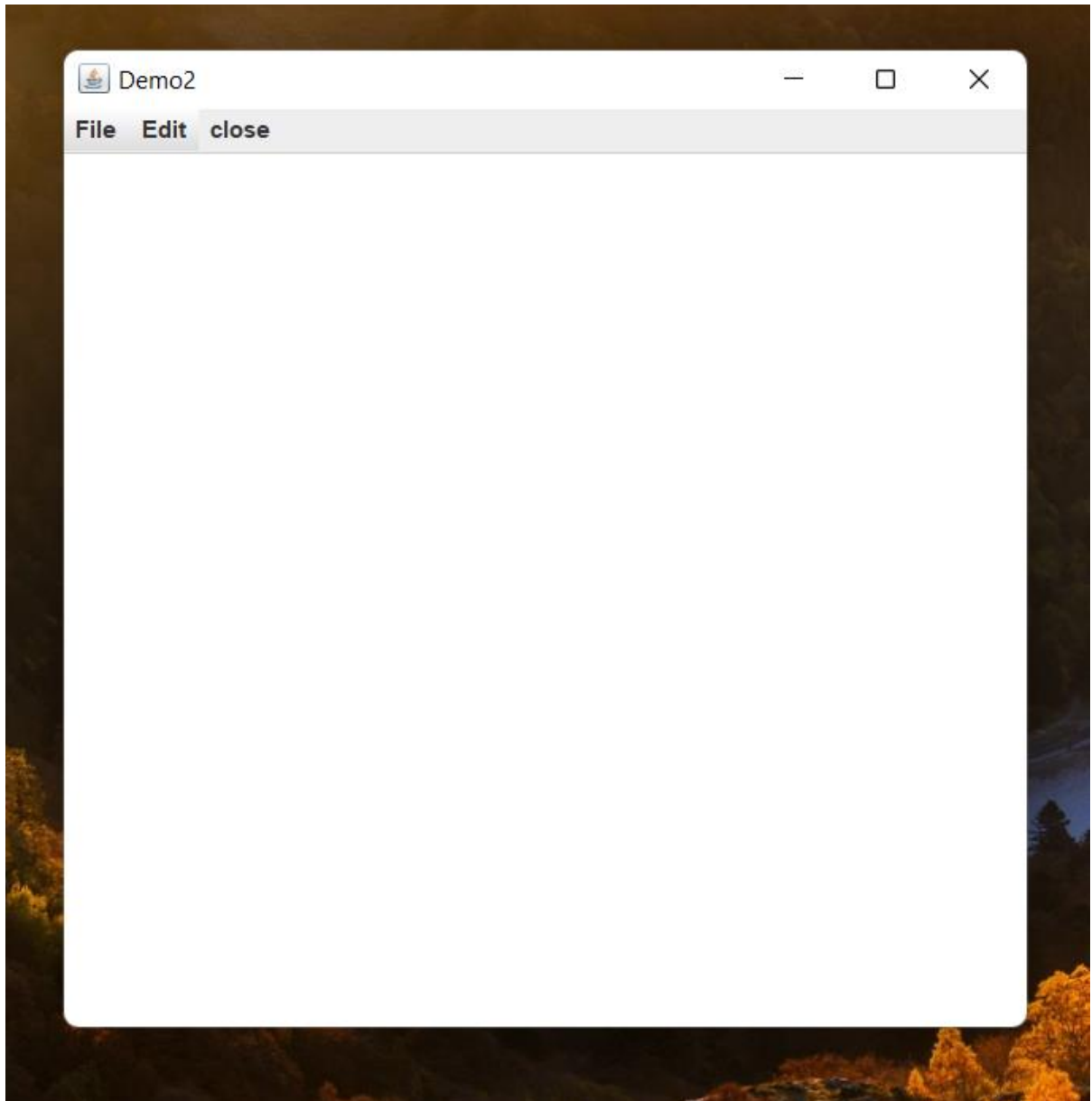


FIG 8

After sharing of Id with first client with another client, multi client communication can be done along with the data in Demo1 can be shared with Demo 2

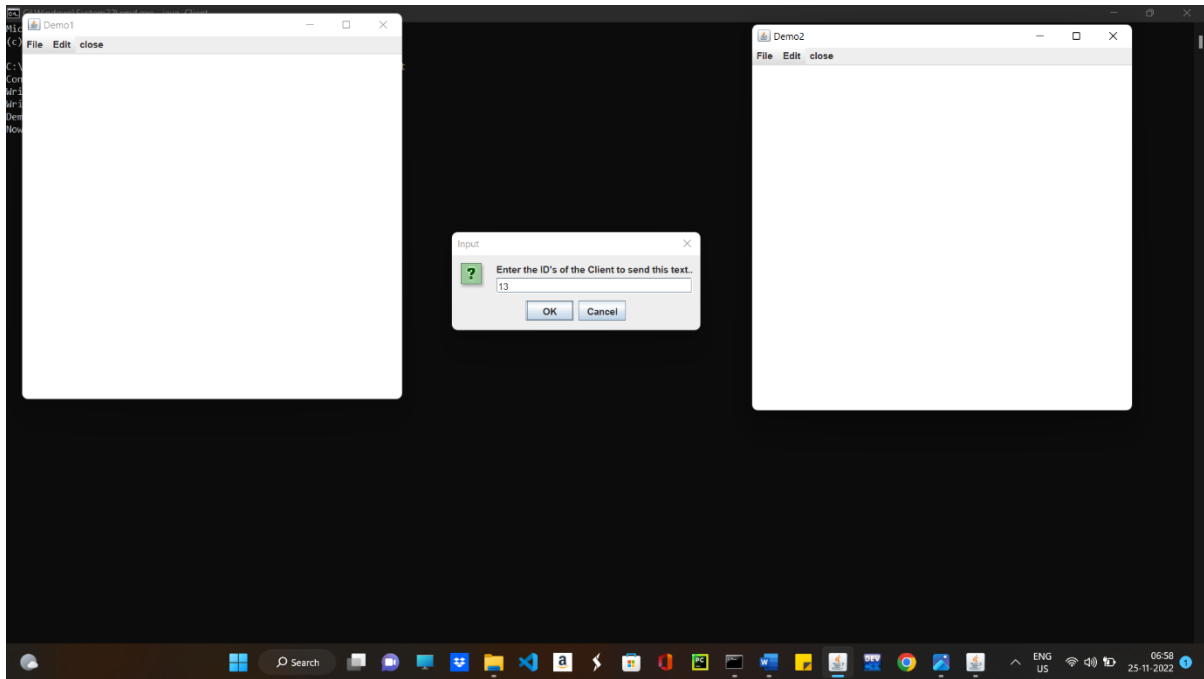


FIG 9

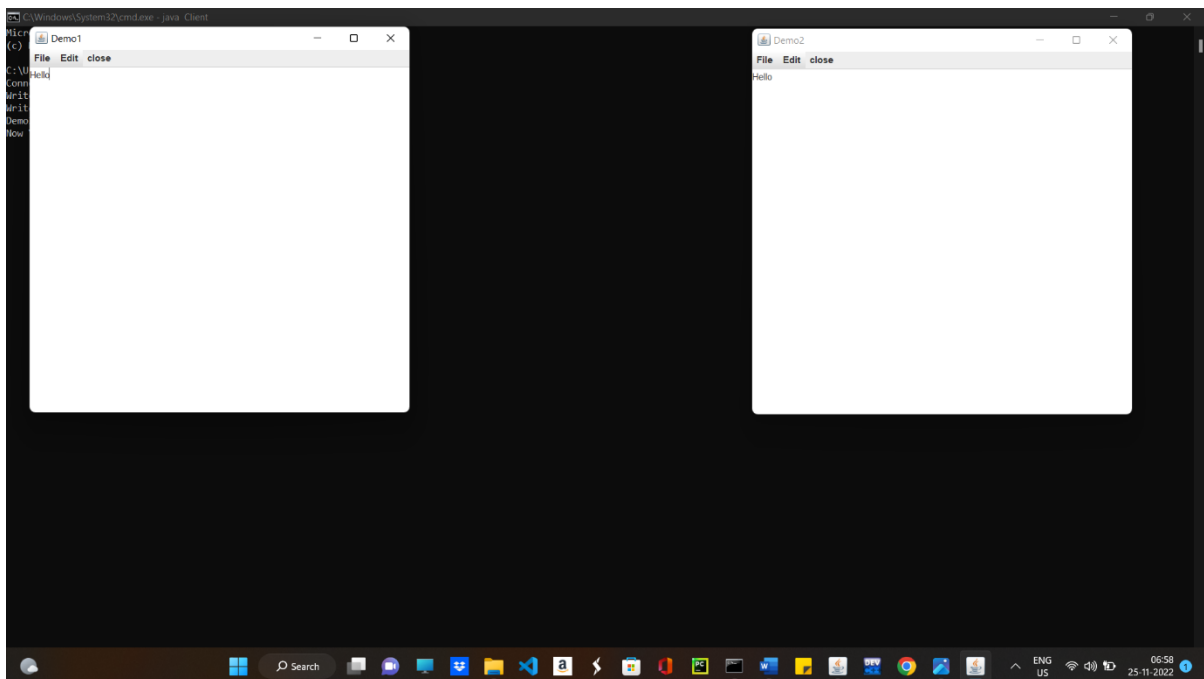


FIG 10

Text can also be saved in the desired location, we can specify the default location where we want to store data. Similarly, we can open an existing file present in the system in the text editor.

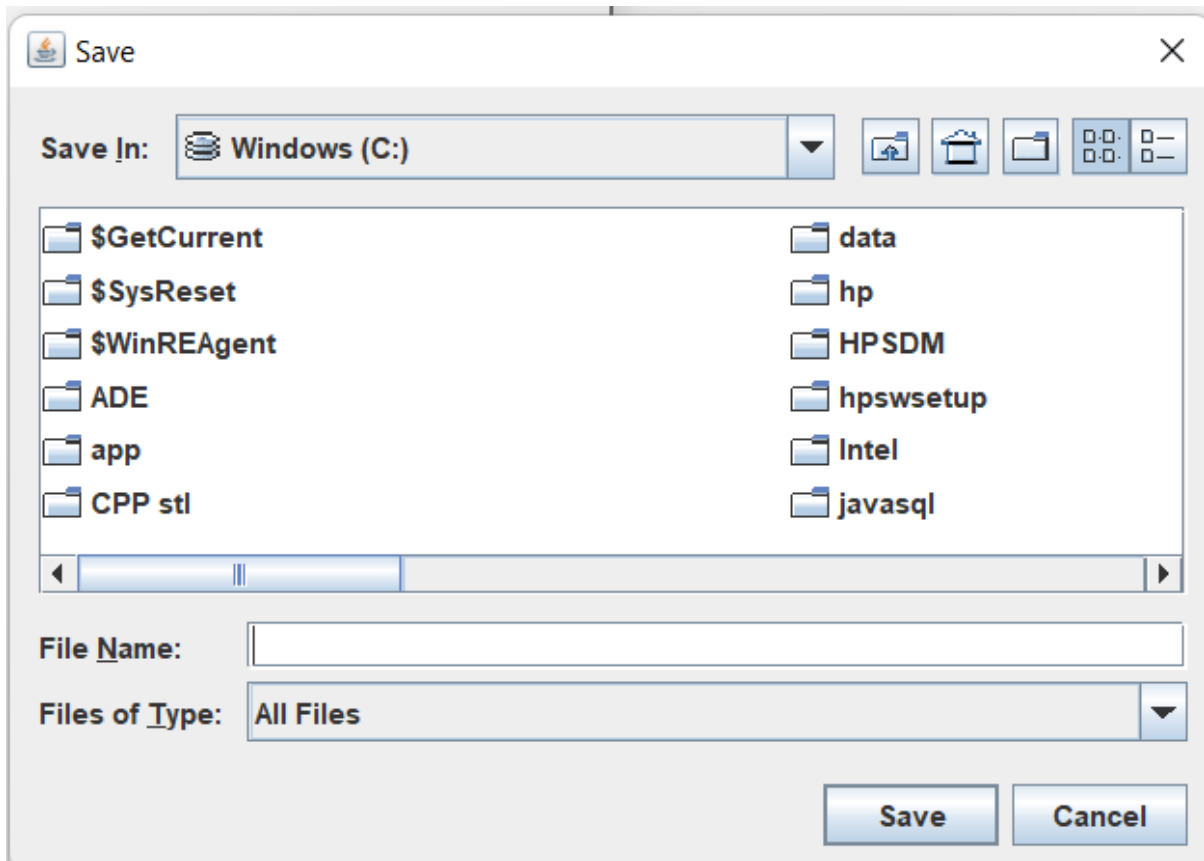


FIG 11

The upload option allows us to share the file between its original destination to specified destination using File Transfer Protocol(FTP).

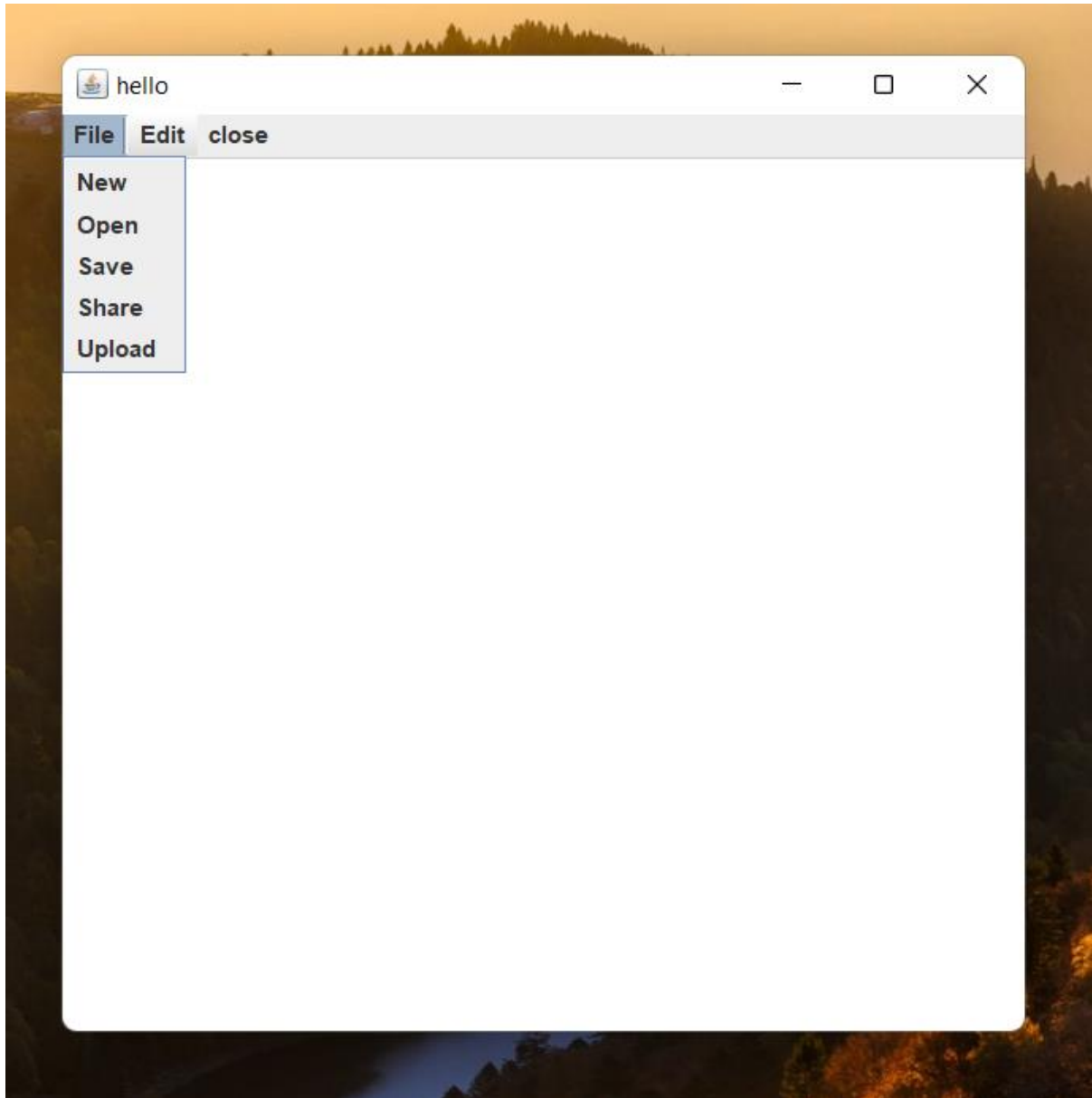


FIG 12

There is also option to Edit(Cut, Copy, Paste) and File(New, Save, Share, Open,Upload)



FIG 13

CHAPTER 4

CONCLUSION AND FUTURE PLANS

This project describes about the concepts and the features of online text editor. The working of online text editor was demonstrated. It briefly explains the client server architecture and the java programming language that is used. This application can still be improved by adding more functionalities so that it looks close to the replica of Google Docs. Insertion of charts, tables and pictures can be implemented in future.

CHAPTER 5

REFERENCES

1. <https://www.javatpoint.com/socket-programming> Javatpoint Socket Programming
2. <https://www.geeksforgeeks.org/socket-programming-in-java/> Geeks for Geeks Socket Programming in Java
3. <https://www.youtube.com/@LearnCodeWithDurgesh> Code by Durgesh Youtube Channel
4. http://gaia.cs.umass.edu/kurose_ross/online_lectures.htm Computer Networking A TOP-DOWN APPROACH-8th EDITION JIM KUROSE, KEITH ROSS AUTHOR'S WEBSITE
5. <https://www.javatpoint.com/computer-network-ftp> Javatpoint FTP
6. Abhijit A. Sawant, Dr. B. B. Meshram / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 3, Issue 1, January -February (2013), pp Network programming in Java using Socket

