# **Remote Desktop App with RDP protocol in Java**

Report submitted to the SASTRA Deemed to be University

as the requirement for the course

# **CSE302: COMPUTER NETWORKS**

submitted by

# **SANJAY RAM G**

**(Reg No:124003275, B.Tech-CSE)**

December 2022

# 

# **SCHOOL OF COMPUTING THANJAVUR,**

**TAMIL NADU, INDIA – 613 401**

 **SCHOOL OF COMPUTING**

**THANJAVUR–613 401**

# **Bonafide Certificate**

# This is to certify that the project work entitled **“Remote Desktop App made with RDP Protocol in Java”** submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Shri/Ms. SANJAY RAM G (Reg No: 124003275, B.Tech CSE)** during the academic year 2022-23, in the School of Computing.

Name Of The Class Teacher : **Dr. KAVITHA**

Project Based Work Viva voce held on

**Examiner - I Examiner - II**

## 

## **ACKNOWLEDGEMENTS**

First of all, I express our gratitude to Prof. Dr S Vaidhyasubramaniam, Vice Chancellor, SASTRA Deemed University, who provided all facilities and constant encouragement during our study. I extend my sincere thanks to Prof. R.Chandra Mouli, Registrar, SASTRA Deemed University for providing the opportunity to pursue this project. It is my privilege to express my sincerest regards to, Prof. Dr A.Uma Maheshwari ,Dean(SOC), Shankar Sriram (Associate Dean (SOC) ) and Dr.N.Kavitha (Computer Networking Staff) who motivated me during the project.

I would specially thank and express my gratitude to my mentor Dr.N.Kavitha (SOC) for her valuable inputs, able guidance, encouragement, wholehearted cooperation and constructive criticism throughout our project on the topic “remote desktop app made with RDP protocol in Java”. I also thank all the Teaching and Non-teaching staff who have directly or indirectly help me through their support, encouragement and all other assistance extended fir completion of my project

**I take this opportunity to thank all lecturers who have directly or indirectly helped my project.**

## **LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 1.1 | RDP Protocol | 9 |
| 1.2 | TCP Protocol | 10 |
| 3.1.1 | Server output 1 | 25 |
| 3.1.2 | Server output 2 | 25 |
| 3.2.1 | Client output 1 | 26 |
| 3.2.2 | Client output 2 | 26 |
| 3.3.1 | Shared Screen | 27 |

# **ABSTRACT**

# 

# 

# The objective of this project is to create a remote desktop app using java language which implements the RDP protocol. Using this project, we are able to share screen and access another laptop remotely.

# RDP allows client to connect with the server. With this protocol, we will be able to run applications on the server from the remote client. RDP is a client-server software package permitting remote network admittance with graphical interface. This application allows us to control the remote server desktop using client’s mouse and keyboard.

**Keywords:** RDP protocol, Client - Server, Java - AWT

**ABBREVIATIONS**

RDP - Remote Desktop Protocol

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

IP - Internet Protocol

GUI - Graphical User Interface

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **NO** | **TITLE** | **PAGE NO** |
| i) | INTRODUCTION | 1 |
| ii) | BONAFIDE CERTIFICATE | 2 |
| iii) | ACKNOWLEDGEMENTS | 3 |
| iv) | LIST OF FIGURES | 4 |
| v) | ABSTRACT | 5 |
| vi) | ABBREVIATIONS | 6 |
| 1 | CHAPTER 1 - INTRODUCTION | 8 |
| 1.1 | NETWORK CONCEPTS | 10 |
| 1.2 | WORKING OF RDP | 11 |
| 1.3 | MERITS AND DEMERITS | 12 |
| 2 | CHAPTER 2 - SOURCE CODE | 13 |
| 3 | CHAPTER 3 - OUTPUT SCREENSHOTS | 25 |
| 4 | CHAPTER 4 - CONCLUSION & FUTURE PLANS | 28 |
| 5 | CHAPTER 5 - REFERENCES | 29 |

# **CHAPTER 1**

# **INTRODUCTION**

RDP means Remote Desktop Protocol. RDP protocol is a protocol which is developed by Microsoft. Remote Desktop Protocol is the facility to connect with the other device and use a faraway system from a separate system. By using this RDP protocol, we can control one system with the help of another system. Remote desktop operators can control their desktop, they can open the files and can also correct the files and use the all apps and all the documents present in the system as if they are actually present at their own system. RDP protocol works by existing operators that permits them to connect to another system remotely.

RDP work with the help of multiple dissimilar types of network technologies. RDP is accessible for almost all the versions of windows OS. RDP provides more security enhanced approach in order to avoid the complications. This app uses verification means that operator’s IP is provided and one can access the other laptop only with the help of the operator IP that is provided. It also provides encryption to avoid the traffic. With the help of this protocol, we are able to run the applications on server system with the help of a client system.

Remote access protocol can also be illuminated as remote regulate of a system through another system linked over the internet or over the personal area network. This is extensively used by countless system constructors and huge businesses help line members for practical problems of their customer’s complications. There are innumerable specialized RDP applications. Remote Desktop Protocol is a tool that is proficient to transmit a system information that is present away from the other system. In another way, it can be expressed as this way, it permits the access on your system and it will be running on other system also. Consider an example, we can use RDP to link one system that is present in office and another system that is present in your home and access the all information and all the display information, all files and all system properties as if we are sitting in the office.

Remote Desktop Organizer is the client or server software file permitting remote network access to another desktop. This RDP software permits us to get the display of the remote system and we are able to control it with our own mouse and keyboard. RDP protocol is helpful to achieve remote system connection. RDP software needs a TCP link among the server and the client. Every system will be having a different IP address that is not same with other IP address and can also have the title in DNS. DNS means Domain Name Server. The one who wants to connect should know the IP address and also it can connect when the name of the server is known. In this TCP connection, the initial handshaking should be present and it involves the Client Initialization and also the Server Initialization communications.

When the linking among a client and a server is first recognized, the server begins by asking for confirmation from the client that normally results in the operator that being urged for a connection from the client side. Then the server and the client interchange message to assign the measurements of the desktop and the format of pixel that should be used.

The server and the client are designed as unpretentious as possible and that is usually up to server to achieve any necessary conversions. Consider an example, the server must give the information of the format of pixel that should be used and also what the client wants. Every desktop is resembling a simulated X display with the help of origin window on that contains numerous X requests could be exhibited. It makes available the key operator of a PC through remote permit to their system. The server handling consists of reclaiming the information of the pixel.

A proper authentication is present in the case of RDP. A user can connect only when he knows that IP address of the system should be known in order to connect the system. So by all these things we can tell that there is no chance of insecurity based approach and it is helpful to almost all the people who are having troubles with their systems. It can be implemented on any software that is also one of the major advantages. There are different programs for accessing different parts of the system and for connection also there are so many programs which are present in the source code of the page.

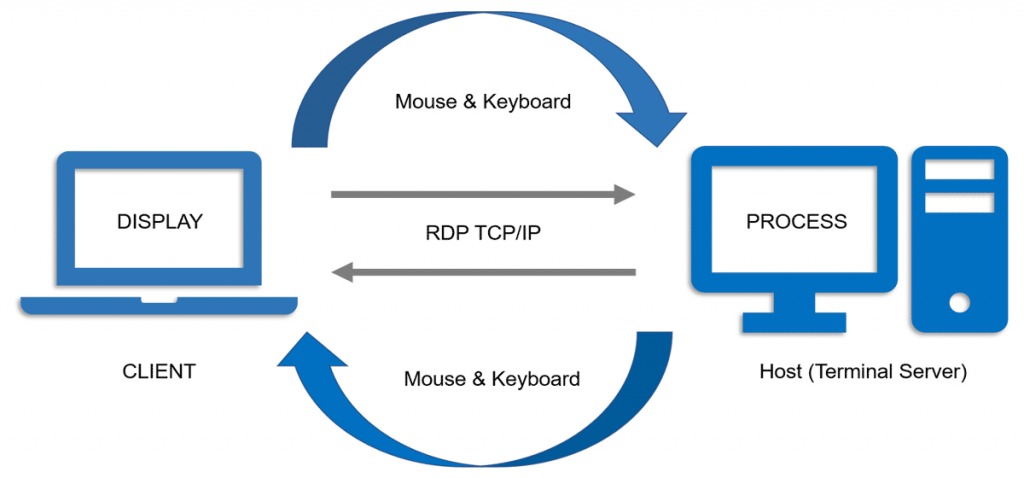
****

Fig 1.1

**NETWORK CONCEPTS**

We have to know some basic concepts and also functionalities regarding implementation of Remote Desktop Protocol. Some of the network concepts which are used in Remote Desktop Controller are as follows.

1. TCP: TCP is abbreviated as Transmission Control Protocol. Transmission Control Protocol gives safe data transmission between the two systems. When two systems want to link to another system consistently, they link together and transmit the required information back and forth with the help of that link. Same as the phone company, Transmission Control Protocol promises that the information transferred from one side of the link really transferred to the another side and also with the same order which has been sent by the sender. Or else, error is informed.

2. SOCKETS: A Socket is one end of a two way interconnection bond considering the two programs executing on the networks. A Socket is assigned to a port number so that the Transmission Control Protocol layer can recognize the request that information is intended to be sent.

3. IP ADDRESS: An Internet Protocol (IP) address is a numerical address assigned to each device connected to a computer network that uses the Internet Protocol for communication. IP address is assigned to your device by your Internet Service Provider, like your cable, telephone, wireless it is a requirement to use the internet. This basic information regarding computer networks is enough to understand this project. Now we will see implementation of project.

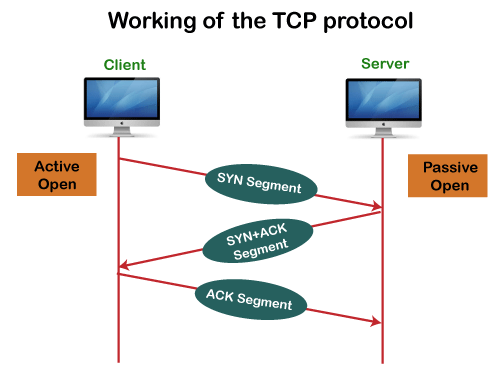
****

Fig 1.2

**WORKING OF RDP PROTOCOL:**

1. To execute this project we need two desktops one will act as a server and another will act as a client. We have several programs in server and client so we will discuss each and every program how they are going to use in both server and client in coding session.

2. First we have to run server program. To run server program first of all we have to setup environment and directories properly. For that in one desktop, first create a folder in any drive and within that folder create one folder and then save all server programs.

3. In another desktop create a folder in any location and within that create one folder and within that save all client programs.

4. Server Side: Now in the server desktop open command prompt and run server programs. Then it will generate one small dialog box and it will ask to start the server. Then you have to click ok in server then it will display that it is waiting for connect to client.

5. Client Side: When we run server programs then server will wait for client request. Now in another desktop we have to run client programs then small dialog box will open. Now we have to enter server IP address in that dialog box. After submitted, a new screen with the screen of the client is displayed.

6. In this way we can execute our programs and we will see how remote desktop controller will work.

**MERITS OF RDP PROTOCOL**:

1. RDP protocol is one of the protocol which is mostly used in our daily life whenever there are troubles facing by us with our devices.

2. It has an easy access. It permits the user to link to the user’s required desktop and the user can access the data by using this protocol.

3. It has low cost. This RDP service reduce the amounts to be invested on the physical desktop systems. It permits the user to work from his own device.

4. Security is the most important feature in the case of RDP protocol.

5. The server device can be very easily updated with fixes that should be done to secure the device. RDP backup the data that is required by the user when the server crashes down.

6. Easy management is possible by the use of this RDP protocol.

7. It has more flexibility. The main purpose of RDP protocol is to permit the workers to perform their tasks from literally at everywhere and at anytime.

8. RDP protocol will be helpful to the people who have troubles with their own systems.

**DEMERITS OF RDP PROTOCOL**:

There are some disadvantages in RDP protocol, some of them are listed below.

* Network dependency occurs in the case of RDP.
* Based on the number of users, performance of the system will be decreased.
* Downtime is one of the disadvantage in the case of RDP protocol.

# **CHAPTER 2**

**SOURCE CODE:**

**SERVER:**

1. **StartServer.java**

package server;

import java.awt.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

public class StartServer {

private JFrame frame = new JFrame();

private JDesktopPane desktop = new JDesktopPane();

public void start(int port){

try {

try (ServerSocket sc = new ServerSocket(port)) {

drawGUI();

while(true){

Socket client = sc.accept();

System.out.println("New client Connected to the server");

new ClientHandler(client,desktop);

}

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

public void drawGUI(){

frame.add(desktop,BorderLayout.CENTER);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

}

public static void main(String args[]){

int ch=JOptionPane.showConfirmDialog(null, "Start Server", "Start", 0);

System.out.println(ch);

if(ch==0){

new StartServer().start(5000);

}

else{

JOptionPane.showMessageDialog(null, "Cancelled", "Cancel", ch);

}

}

}

1. **ScreenReceiver.java**

package server;

import java.awt.\*;

import java.io.\*;

import javax.swing.\*;

class ScreenReciever extends Thread {

private ObjectInputStream cObjectInputStream = null;

private JPanel cPanel = null;

private boolean continueLoop = true;

public ScreenReciever(ObjectInputStream ois, JPanel p) {

cObjectInputStream = ois;

cPanel = p;

start();

}

public void run(){

try {

while(continueLoop){

ImageIcon imageIcon = (ImageIcon) cObjectInputStream.readObject();

System.out.println("New image recieved");

Image image = imageIcon.getImage();

image = image.getScaledInstance(cPanel.getWidth(),cPanel.getHeight(),Image.SCALE\_FAST);

Graphics graphics = cPanel.getGraphics();

graphics.drawImage(image, 0, 0, cPanel.getWidth(),cPanel.getHeight(),cPanel);

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

}

1. **ActionSender.java**

package server;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

class ActionSender implements KeyListener,MouseMotionListener,MouseListener {

private Socket cSocket = null;

private JPanel cPanel = null;

private PrintWriter writer = null;

private Rectangle clientScreenDim = null;

ActionSender(Socket s, JPanel p, Rectangle r) {

cSocket = s;

cPanel = p;

clientScreenDim = r;

cPanel.addKeyListener(this);

cPanel.addMouseListener(this);

cPanel.addMouseMotionListener(this);

try {

writer = new PrintWriter(cSocket.getOutputStream());

} catch (IOException ex) {

ex.printStackTrace();

}

}

public void mouseMoved(MouseEvent e) {

double xScale = clientScreenDim.getWidth()/cPanel.getWidth();

System.out.println("xScale: " + xScale);

double yScale = clientScreenDim.getHeight()/cPanel.getHeight();

System.out.println("yScale: " + yScale);-

System.out.println("Mouse Moved");

writer.println(Commands.MOVE\_MOUSE.getAbbrev());

writer.println((int)(e.getX() \* xScale));

writer.println((int)(e.getY() \* yScale));

writer.flush();

}

public void mousePressed(MouseEvent e) {

System.out.println("Mouse Pressed");

writer.println(Commands.PRESS\_MOUSE.getAbbrev());

int button = e.getButton();

int xButton = 16;

if (button == 3) {

xButton = 4;

}

writer.println(xButton);

writer.flush();

}

public void mouseReleased(MouseEvent e) {

System.out.println("Mouse Released");

writer.println(Commands.RELEASE\_MOUSE.getAbbrev());

int button = e.getButton();

int xButton = 16;

if (button == 3) {

xButton = 4;

}

writer.println(xButton);

writer.flush();

}

public void keyPressed(KeyEvent e) {

System.out.println("Key Pressed");

writer.println(Commands.PRESS\_KEY.getAbbrev());

writer.println(e.getKeyCode());

writer.flush();

}

public void keyReleased(KeyEvent e) {

System.out.println("Mouse Released");

writer.println(Commands.RELEASE\_KEY.getAbbrev());

writer.println(e.getKeyCode());

writer.flush();

}

public void mouseEntered(MouseEvent e) {

}

public void mouseExited(MouseEvent e) {

}

public void keyTyped(KeyEvent e) {

}

public void mouseClicked(MouseEvent e) {

}

public void mouseDragged(MouseEvent e) {

}

}

1. **ClientHandler.java**

package server;

import java.awt.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

class ClientHandler extends Thread {

private JDesktopPane desktop = null;

private Socket cSocket = null;

private JInternalFrame interFrame = new JInternalFrame("Client Screen",false, false, false);

private JPanel cPanel = new JPanel();

public ClientHandler(Socket cSocket, JDesktopPane desktop) {

((javax.swing.plaf.basic.BasicInternalFrameUI)interFrame.getUI()).setNorthPane(null);

this.cSocket = cSocket;

this.desktop = desktop;

start();

}

public void drawGUI(){

interFrame.setLayout(new BorderLayout());

interFrame.getContentPane().add(cPanel,BorderLayout.CENTER);

interFrame.setSize(100,100);

desktop.add(interFrame);

try {

interFrame.setMaximum(true);

}catch (Exception ex) {

ex.printStackTrace();

}

cPanel.setFocusable(true);

interFrame.setVisible(true);

}

public void run(){

Rectangle clientScreenDim = null;

ObjectInputStream ois = null;

drawGUI();

try{

ois = new ObjectInputStream(cSocket.getInputStream());

clientScreenDim =(Rectangle) ois.readObject();

}catch(IOException ex){

ex.printStackTrace();

}catch(ClassNotFoundException ex){

ex.printStackTrace();

}

new ScreenReciever(ois,cPanel);

new ActionSender(cSocket,cPanel,clientScreenDim);

}

}

1. **Commands.java**

package server;

public enum Commands {

PRESS\_MOUSE(-1),

RELEASE\_MOUSE(-2),

PRESS\_KEY(-3),

RELEASE\_KEY(-4),

MOVE\_MOUSE(-5);

private int abbrev;

Commands(int abbrev){

this.abbrev = abbrev;

}

public int getAbbrev(){

return abbrev;

}

}

**CLIENT:**

1. **StartClient.java**

package client;

import java.awt.\*;

import java.awt.event.\*;

import java.net.\*;

import javax.swing.\*;

public class StartClient {

Socket socket = null;

public void initialize(String ip, int port ){

Robot robot = null;

Rectangle rectangle = null;

try {

System.out.println("Connecting to server ..........");

socket = new Socket(ip, port);

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

GraphicsEnvironment gEnv=GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice gDev=gEnv.getDefaultScreenDevice();

Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

rectangle = new Rectangle(dim);

robot = new Robot(gDev);

drawGUI();

new ScreenSender(socket,robot,rectangle);

new ActionReciever(socket,robot);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void drawGUI() {

JFrame frame = new JFrame("Remote Admin");

JButton button= new JButton("Terminate");

frame.setBounds(100,100,150,150);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.add(button);

button.addActionListener( new ActionListener() {

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

frame.setVisible(true);

}

public static void main(String[] args){

String ip = JOptionPane.showInputDialog("Please enter server address");

new StartClient().initialize(ip, 5000);

}

}

1. **ScreenSender.java**

package client;

import java.awt.\*;

import java.awt.image.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

class ScreenSender extends Thread {

Socket socket = null;

Robot robot = null;

Rectangle rectangle = null;

boolean continueLoop = true;

public ScreenSender(Socket socket, Robot robot,Rectangle rect) {

this.socket = socket;

this.robot = robot;

rectangle = rect;

start();

}

public void run(){

ObjectOutputStream oos = null;

try{

oos = new ObjectOutputStream(socket.getOutputStream());

oos.writeObject(rectangle);

}catch(IOException ex){

ex.printStackTrace();

}

while(continueLoop){

BufferedImage image = robot.createScreenCapture(rectangle);

ImageIcon imageIcon = new ImageIcon(image);

try {

System.out.println("before sending image");

oos.writeObject(imageIcon);

oos.reset();

System.out.println("New screenshot sent");

} catch (IOException ex) {

ex.printStackTrace();

}

try{

Thread.sleep(100);

}catch(InterruptedException e){

e.printStackTrace();

}

}

}

}

1. **ActionReciver.java**

package client;

import java.awt.\*;

import java.io.\*;

import java.net.\*;

import java.util.\*;

class ActionReciever extends Thread {

Socket socket = null;

Robot robot = null;

boolean continueLoop = true;

public ActionReciever(Socket socket, Robot robot) {

this.socket = socket;

this.robot = robot;

start();

}

public void run(){

Scanner scanner = null;

try {

System.out.println("Preparing InputStream");

scanner = new Scanner(socket.getInputStream());

while(continueLoop){

System.out.println("Waiting for command");

int command = scanner.nextInt();

System.out.println("New command: " + command);

switch(command){

case -1:

robot.mousePress(scanner.nextInt());

break;

case -2:

robot.mouseRelease(scanner.nextInt());

break;

case -3:

robot.keyPress(scanner.nextInt());

break;

case -4:

robot.keyRelease(scanner.nextInt());

break;

case -5:

robot.mouseMove(scanner.nextInt(), scanner.nextInt());

break;

}

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

1. **Commands.java**

package client;

public enum Commands {

PRESS\_MOUSE(-1),

RELEASE\_MOUSE(-2),

PRESS\_KEY(-3),

RELEASE\_KEY(-4),

MOVE\_MOUSE(-5);

private int abbrev;

Commands(int abbrev){

this.abbrev = abbrev;

}

public int getAbbrev(){

return abbrev;

}

}

# **CHAPTER 3**

**OUTPUT SCREENSHOTS:**

**server:**

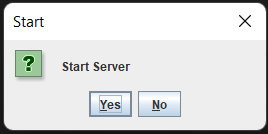


Fig 3.1.1

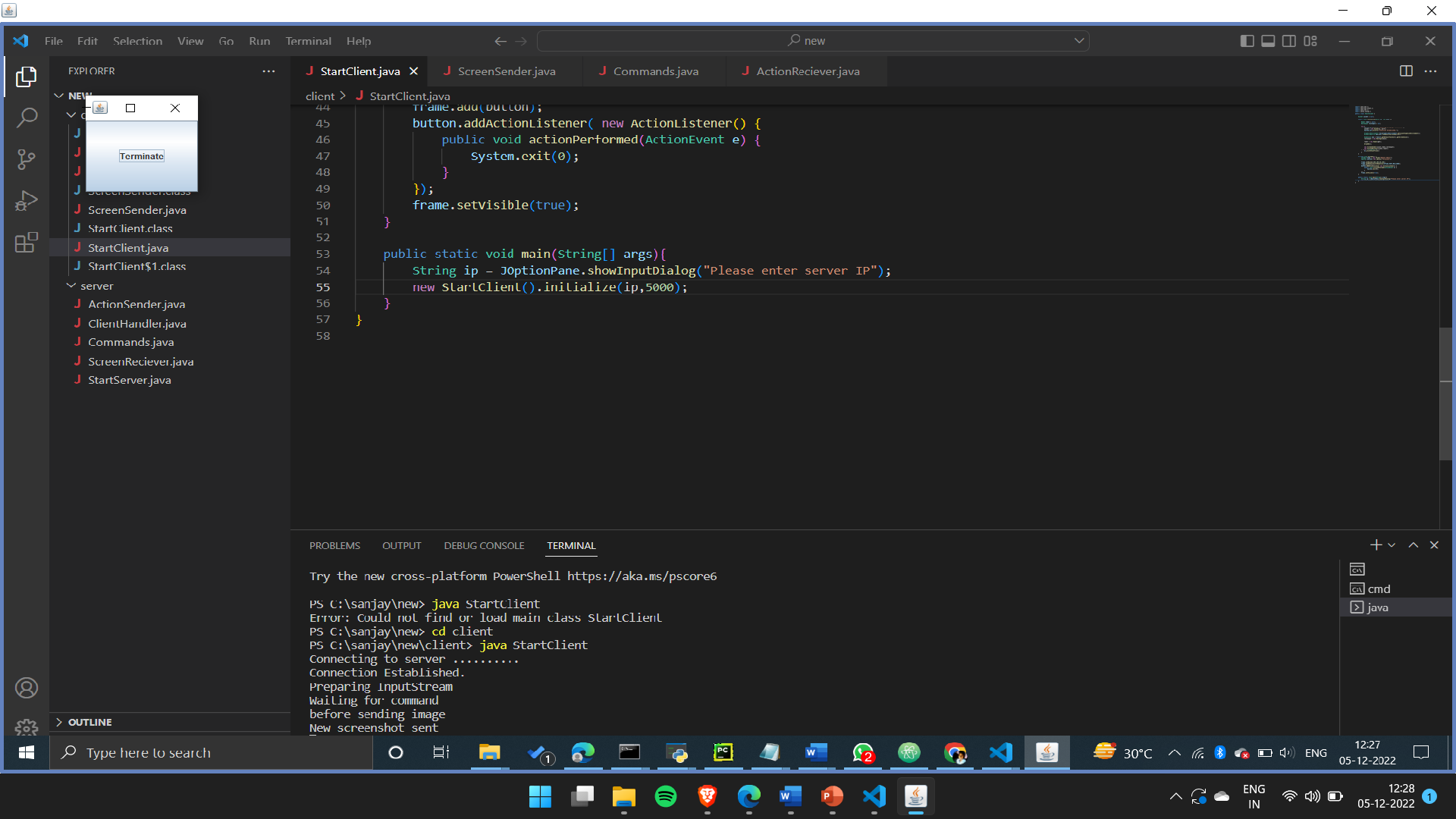


Fig 3.1.2

**client:**

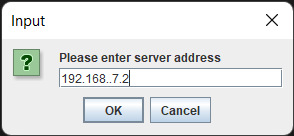


Fig 3.2.1

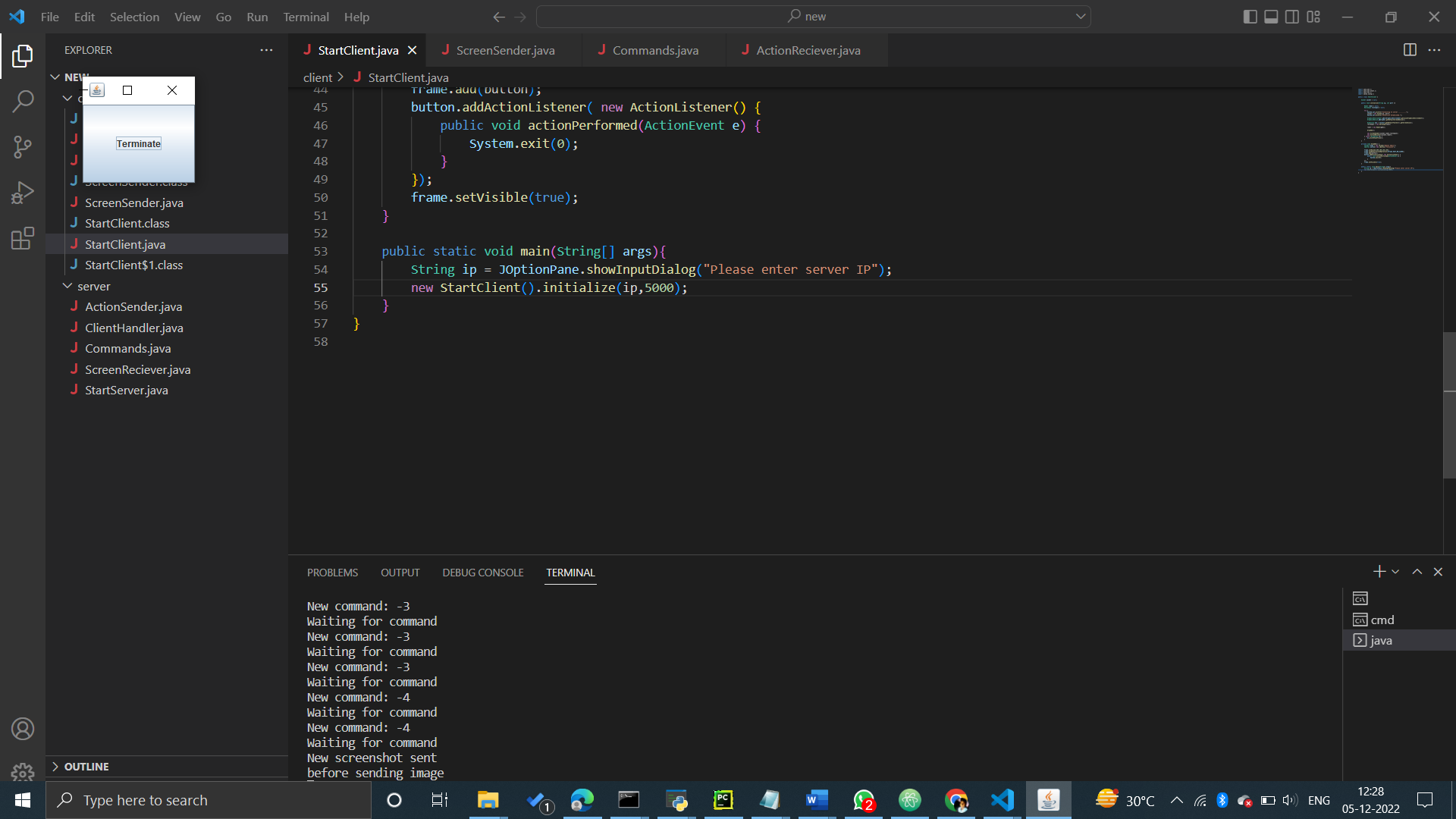


Fig 3.2.2

**Screen Shared and can be controlled :**



Fig 3.3.1

# **CHAPTER 4**

**CONCLUSION:**

* RDP protocols after checking, it has been establish that it has achieved the desired output.
* Implementation of both protocols has been tested with real information and it has been found to be with no error.
* It is also found that the system has been working effectively after implementing the protocols

**FUTURE PLANS:**

Since the Technology has been updating so quickly and also by considering the location of networking the manager has to manage the total things. RDP has vast capability over the Internet based architectures and also has vast capability over the service oriented architectures. To subsist as of that race, each and every system have to do some changes to that in the future. New modifications will give the system a good look so that it can fascinate new many administrators.

Because of this reason, it is a requirement that the system has to be changed based on the users needs. Some of the future improvements is

1. The current usefulness mainly meant for environments of intranet, for example virtual classrooms is also comes under huge range of application areas of intranet environments.

2. Next generation of value package hypothetical on the internet-based applications.

3. Encrypting the data shared to other system before sharing and decrypting it to avoid cyber security issues.

4. Additionally, differences target to take account of abilities such as controls and extra operator controls on the Remote systems.

# **CHAPTER 5**

# **REFERENCES**:

1.<https://en.wikipedia.org/wiki/Remote_Desktop_Protocol>

2.<https://www.parallels.com/blogs/ras/rdp-meaning/>

3.<https://www.javatpoint.com/computer-network-ftp>

4.<https://computingforgeeks.com/how-to-enable-remote-desktop-protocol-rdp-on-windows-server-2019/>

5.<https://searchenterprisedesktop.techtarget.com/definition/Remote-Desktop-Protocol-RDP/>

6.<https://www.howtogeek.com/175087/how-to-enable-and-secure-remote-desktop-on-windows/>