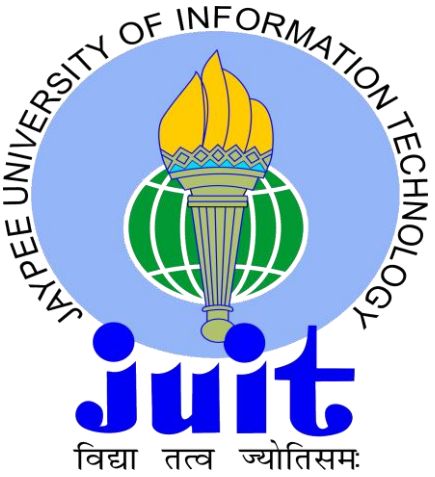
ONLINE SCREEN SHARING

Enrol. No.	081402 081404 081408 081418
Name of Student	Rohit Soni Anurag Kakkar Shivam Gupta Zeeshan Hussain
Name of supervisor(s)	Dr. Ravi Rastogi



May – 2012

Submitted in partial fulfillment of the Degree of
Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
WAKNAGHAT

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the work titled "**Online Screen Sharing**" submitted by "**Rohit Soni, Anurag Kakkar, Shivam Gupta, Zeeshan Hussain**" in partial fulfillment for the award of degree of B.Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor ……..

Name of Supervisor ……..

Designation ……..

Date ……..

ACKNOWLEDGEMENT

We owe a great thanks to all the people who helped us throughout this project duration and make it a worthwhile experience.

Our deepest thanks to project guide Dr.Ravi Rastogi for guiding and leading us throughout the duration of project.He has taken pain to go through the project and make necessary corrections as and when needed.

We express our thanks to HOD sir Brig.S. P. Ghrera for extending his support.

We would also like to thank my institution and my faculty members without whom this project would have been a distant reality.We would also like to extend our heartfelt thanks to our family.

……..	……..	……..	……..
Shivam Gupta	Rohit Soni	Anurag Kakkar	Zeeshan Hussain

Date:28 May 2012

SUMMARY

Our aim is to develop a screen sharing system in which client's screen or client's system is controlled by server. To accomplish our aim we have created an online screen sharing system by using **JAVA** as front end. The desktop sharing software works by mimicking the display of remote computer, the host, on the display of the connecting computer, the client. It does this by copying the client's display and redrawing it on the hosts display. Keyboard and mouse gestures from the client are transmitted to the host, where the machine interprets them as if they had been inputted locally. Because the service only needs to send small amounts of information (keyboard, mouse, and display) over the network and often takes advantage of compression technologies, it therefore works well in low-bandwidth scenarios. Once connected to a remote machine through a remote desktop connection, you have access to all of its applications and data. PC remote control is a vital feature for remote desktop solutions, as it enables the users to administer online computer support . Tech support departments in particular benefit immensely from the flexibility they gain by being able to respond quickly to server and workstation problems occurring anywhere within the network from any computer to which they have remote desktop access .

_____	_____
Signature of Student	Signature of Supervisor
Name	Name: Mr Ravi Rastogi
Date	Date

Chapter 1

INTRODUCTION

The term online screen sharing refers to a software or an operating system feature allowing applications, either command line programs or graphical applications, to be run remotely on a server, while being displayed locally. Screen sharing or remote desktop applications have varying features. Some allow attaching to an existing user's session and “remote controlling”. Taking over a desktop remotely is a form of remote administration. Screen sharing allows you to show a live view of your computer screen to anyone else on the Internet. It allows remote observance or control of any machine on the local network that has Screen Sharing activated. The user is provided with a user-friendly graphical interface for remotely accessing all of the remote computer’s applications, files, and network resources. These resources become available to the user as if they were directly in front of the remote workstation. The server displays a copy of the image received from the client's display screen. The copy is updated on a time interval. The software on server transmits its own keyboard and mouse activity to the client, where the remote control software implements these actions. The client computer behaves as if the actions were performed directly at that computer. This is widely used by many computer manufacturers and large businesses help desks for technical troubleshooting of their customers’ problems.

PURPOSE OF PROJECT

Screen sharing is the ability of the client to share any application or window from their own computer and show it the server. Remote desktop technology offers many benefits for businesses and to people who want access to their office computer while at home, or to their home computer while at the office, or access to either while travelling. The effect this technology has had in the business world has been dramatic. As the stability of the Internet has grown, remote access has matured from convenience to necessity for most businesses. Tech support departments in particular benefit immensely from the flexibility they gain by being able to respond quickly to server and workstation problems occurring anywhere within the network from any computer to which they have remote desktop access.Screen sharing will make life convenient and easy, any number of clients can be connected to a particular server at a time and can solve their problem.

OBJECTIVES OF PROJECT

1) The main objective of the online screen sharing Java project is to design a Remote Administration environment in computer networks.
2) Desktop sharing software provides support to its entire client over the network.
3) Using this application we can share our desktop all over the world.
4) This can behave as a network administrator to its client to provide remote services like remote messaging, remote file transfer and remote software installation, remote shutdown, remote restart, remote logoff, remote desktop sharing and remote chatting.
5) It should work on all windows platform and written in java.
6) It should be cost effective and should utilize few kbs of memory space.
7) Since it is cheap and efficient so it should have a good market penetration.

Chapter-2

SOFTWARE DEVELOPMENT LIFE CYCLE

\

Software Development Life Cycle is a structured imposed on the development of a software product. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Several steps involve in development of software are :-

1) **Project Planning** : Determines the project's goals and results in a high- level view of the potential project. A feasibility study may be undertaken as part of this phase.
2) **Requirement Analysis** : Results in the creation of well-defined functions from the defined project goals.
3) **Software Design** : Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudocode and other documentation.
4) **Software Coding** : The analyst translates the code or the programs in such a way that they become machine readable form.
5) **Software Testing** : Brings all the pieces of code together into a special testing environment, then checks for errors, bugs and interoperability.
6) **Depolyment** : The final stage of initial development, where the software is put into production and runs actual business.This is a vital stage as analyst waits for positive feedback.
7) **Maintenance** : The last stage of the SDLC is that the analyst needs to maintain the system and see to it that it working within the standards set. He needs to maintain the system by removing the defects of flaws occured.

SOFTWARE DEVELOPMENT MODEL USED

Structured Evolutionary Prototyping Model

It refers to the activity of creating prototypes of software applications i.e. incomplete versions of the software program being developed. A prototype typically simulates only a few aspects of ,and may be completely different from the final product. Steps involved in this model are :-

1) A preliminary project plan is developed.
2) An partial high-level paper model is created.
3) The model is source for a partial requirements specification.
4) A prototype is built with basic and critical attributes.
5) We can demonstrate the prototype and evaluated for problems and made improvements.
6) This loop continues until we get satisfied.

Strengths

1) We can see the system requirements as they are being gathered.
2) We learned from various prototypes made.
3) A more accurate end product.
4) It allows for flexible design and development.
5) Visible signs of progress produced.
6) Our interaction with various prototypes simulates additional needed functionality.

Weakenesses

1) This prototype takes time and no space for bad and dirty methods.
2) Overall improvement of software may be overlooked.
3) Process may continue forever since every prototype stage may require a new functionality to be added.

FEASIBILITY STUDY

Feasibility study aims to objectively and rationally uncover the strengths and weaknesses of the existing software, opportunities and threats as presented by the environment and ultimately the prospects for success .In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. Generally, feasibility study precede technical development and project implementation. Feasibility study concentrates on following areas :-

1) **Technical feasibility** : Since the software is based on simple java.The various features of java that are being used are swing, AWT, socket programming and we are acquainted with this language very well. So the software is fairly feasible to develop after enough research.

2) **Operational Feasibility** : Since the software is easy to install and uses less RAM memory of 256 Mb.It works only on windows based platform and easy to use and has a simple interface to work on so it has high usability and can work effectively in large organisations.

3) **Economic Feasibility** : We believe in open source software and kept it cost free for users.

Project	Technology	Server	Client	Multiple Sessions	Encryption	Authentication	Image Quality	FILE TRANSFER	CHAT
AJAX Remote Desktop Viewer	Socket	✓	✓	✓	X	X	X	X	X
Dayon!	Socket	✓	✓	X	X	X	✓	X	X
Java Remote Desktop	RMI	✓	✓	✓	✓	✓	✓	✓	X
VNC Viewer	Socket	X	✓	✓	X	✓	✓	✓	✓
Online Screen Sharing	Socket	✓	✓	✓	X	✓	✓	X	X

Comparison table for various Java remote desktop applications.

SOFTWARE REQUIREMENT

NetBeans:

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment(IDE) for developing with Java, JavaScript, PHP, Python, Groovy, C, C++ and others. The NetBeans IDE is written in Java Coffee and can run anywhere a JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality. The **NetBeans Platform** is a reusable framework for simplifying the development of Java Swing desktop applications. Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)
- NetBeans Visual Library
- Integrated development tools

NetBeans IDE is a free, open-source, cross-platform IDE with built-in-support for java programming language.

HARDWARE REQUIREMENT

Operating System: Windows XP and above

RAM: 160 MB(minimum)

Hard Disk: 80 GB(minimum)

Lanwire (for local area connection)

LANGUAGE USED

JAVA :

Java is a programming language originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java is currently one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

Java Platform :

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code. Java bytecode instructions are analogous to machine code, but are intended to be interpreted by a virtual machine written specifically for the host hardware. End-users commonly use a Java Runtime Environment(JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

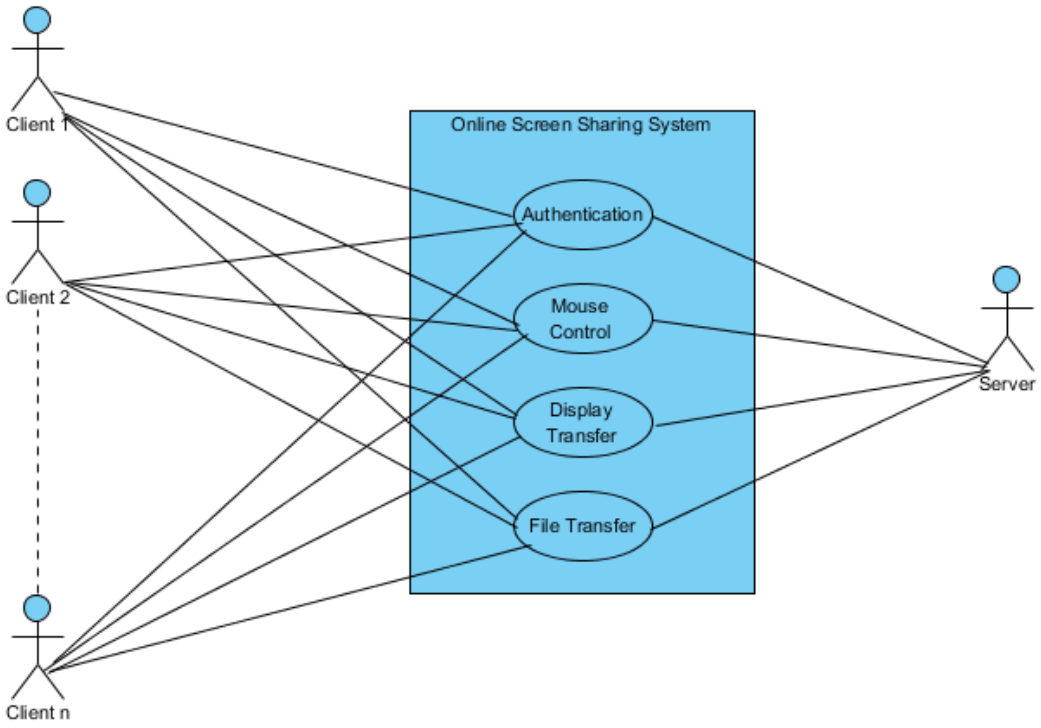
Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time (JIT) compilers were introduced from an early stage that compile bytecodes to machine code during runtime.

Chapter 3

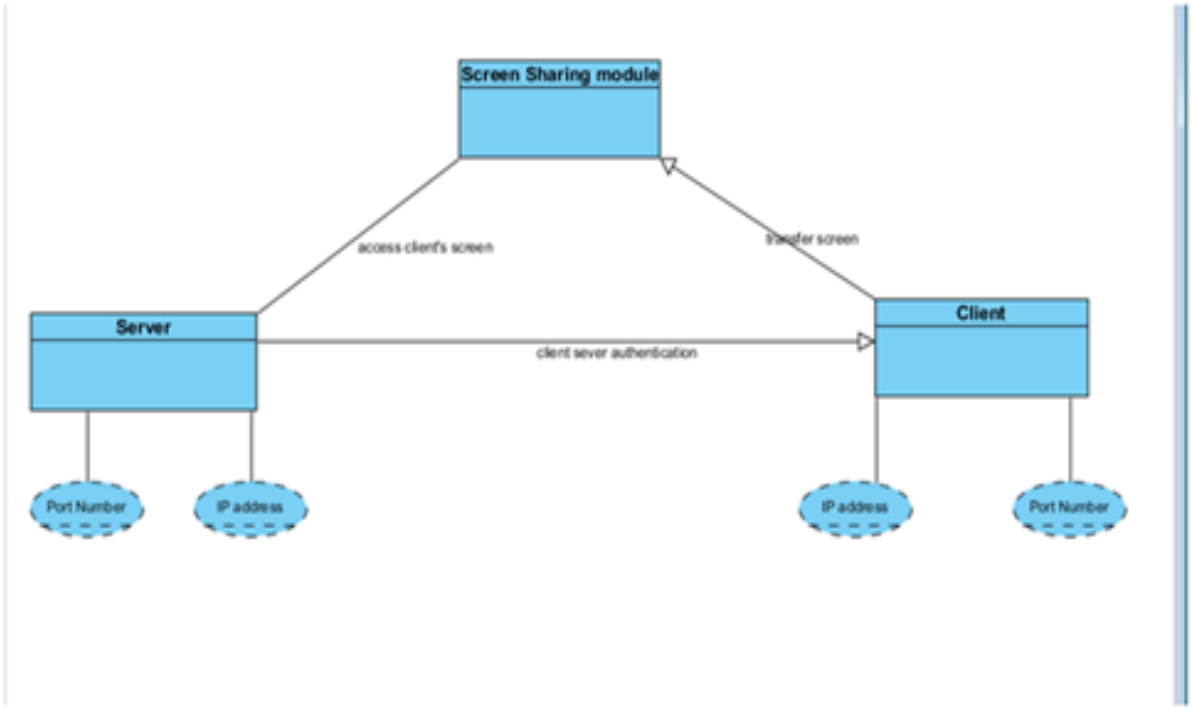
SYSTEM DESIGN

UseCaseDiagram :



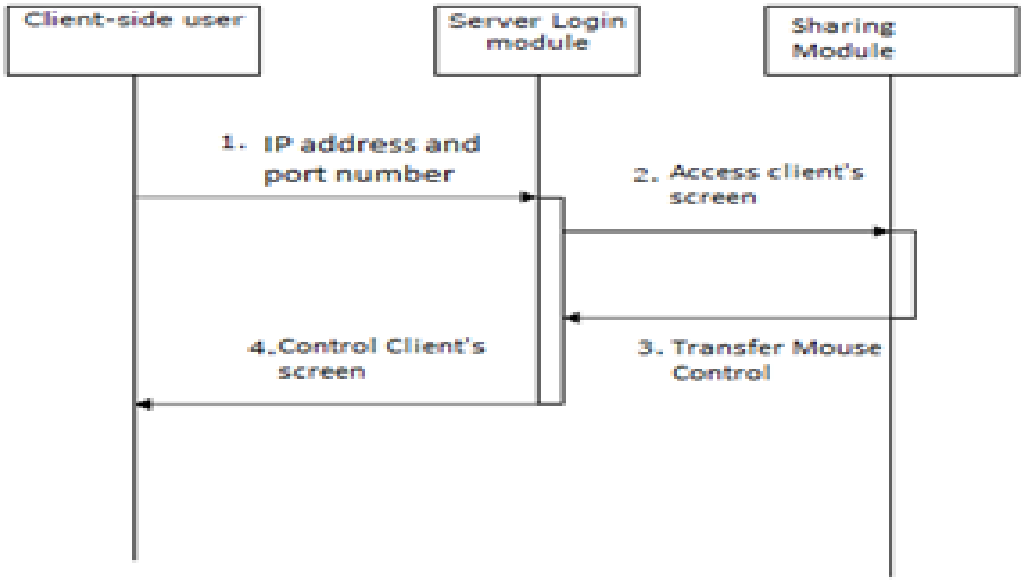
A Use Case diagram is a diagram which shows interaction between various users, also known as Actors and various functionalities provided by a software, also known as Use Cases.

Class diagram :



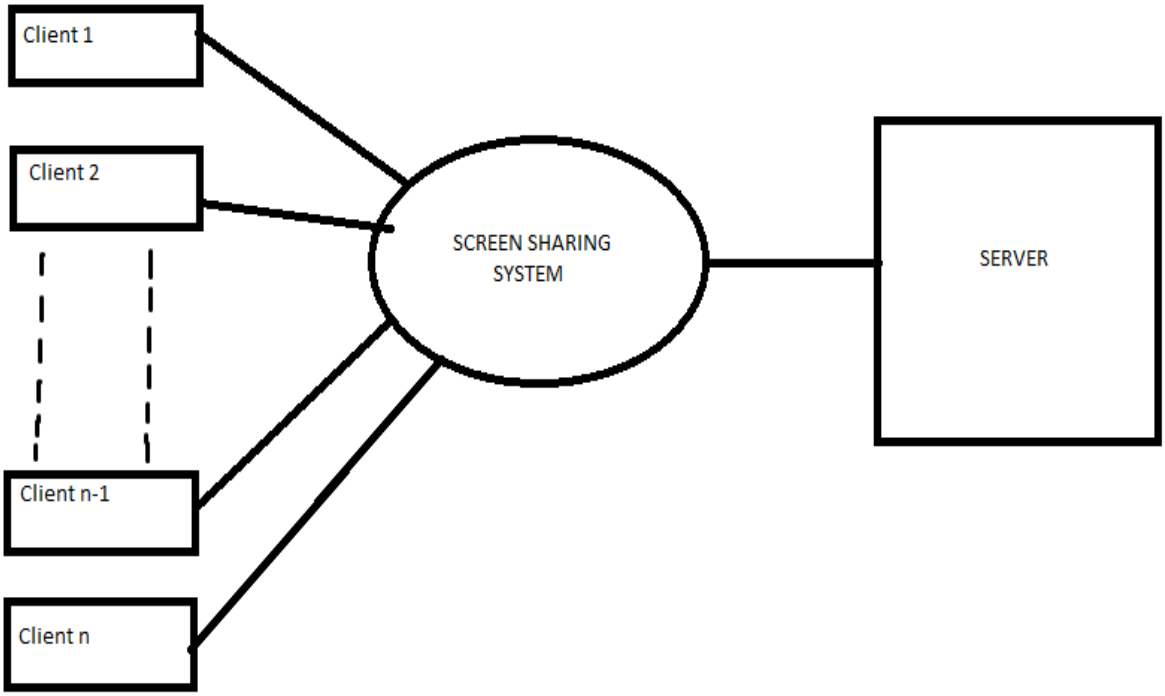
A Class Diagram is a diagram representing the various structure of a system by showing them as classes, their attributes, operations and relationships among the classes.

Sequence Diagram :



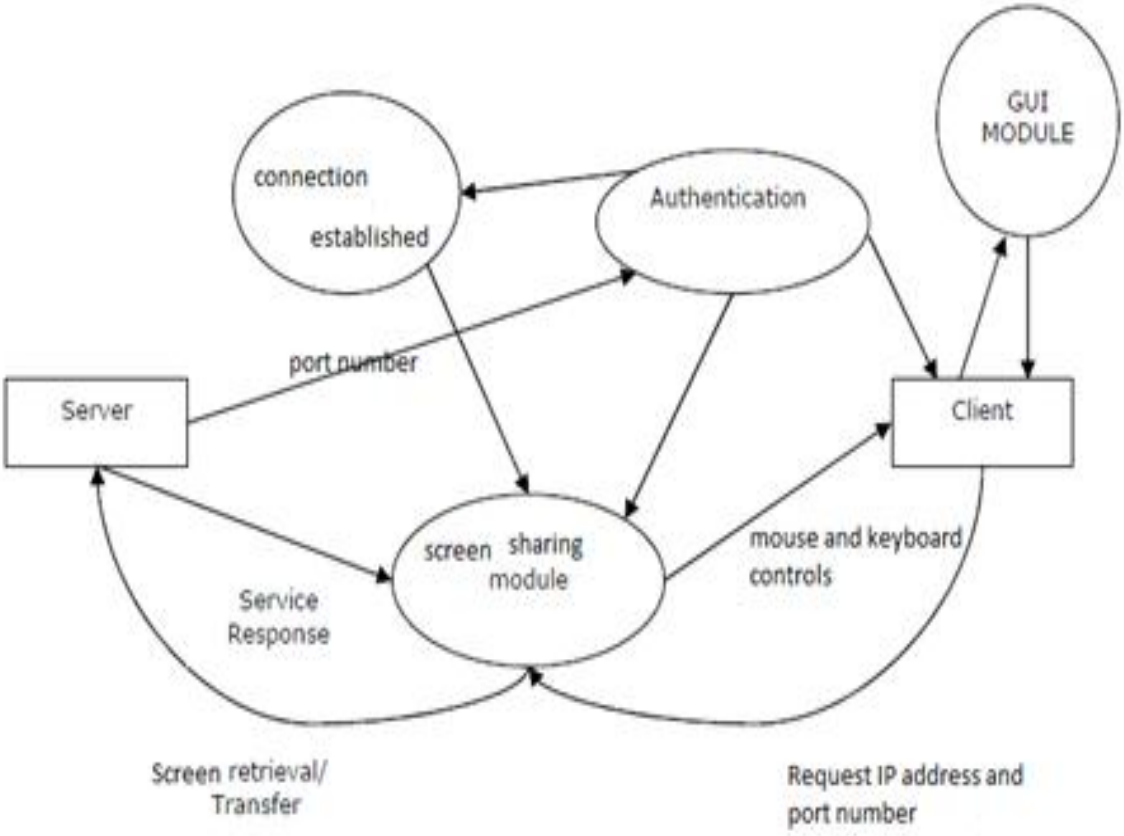
A sequence diagram is a digram which shows how processes operate with one another. A Sequence diagram shows interaction between different objects arranged in time sequence. It represents the objects and classes involved in the scenario and the sequence of messages exchange between different objects to carry out the functionality of the scenario. Sequence diagram depicts the logical view of system under development.

DFD level 0 :



Data Flow Diagram is the graphical representation of flow of data. It is a preliminary step used to create the overview of system that can be elaborated later on. DFD can also be used for structure design and data processing. A DFD describes the type of input and output given to the system and from where this data come, go and where the data will be stored. It does not show whether the processes will operate in a timely manner.

DFD level 1 :


GUI
MODULE
connection
established
Authentication
port number
Server
Client
screen sharing
module
mouse and keyboard
controls
Service
Response
Screen retrieval/
Transfer
Request IP address and
port number

CODING

Client Remote Control:

1. Client Initiator Class

```
packageremoteclient;

importjava.awt.AWTException;
importjava.awt.Dimension;
importjava.awt.GraphicsDevice;
importjava.awt.GraphicsEnvironment;
importjava.awt.Rectangle;
importjava.awt.Robot;
importjava.awt.Toolkit;
importjava.awt.event.ActionEvent;
importjava.awt.event.ActionListener;
importjava.io.IOException;
importjava.net.Socket;
importjava.net.UnknownHostException;
importjavax.swing.JButton;
importjavax.swing.JFrame;
importjavax.swing.JOptionPane;

/*
 * This class is responsible for connecting to the server
 * and starting ScreenSpyer and ServerDelegate classes
 */
public class ClientInitiator {

    Socket socket = null;
   public static void main(String[] args)
```

```
      {
       String ip = JOptionPane.showInputDialog("Please enter server IP");
       String port = JOptionPane.showInputDialog("Please enter server port");
       newClientInitiator().initialize(ip, Integer.parseInt(port));
   }

public void initialize(String ip, int port )
      {
       Robot robot = null; //Used to capture the screen
       Rectangle rectangle = null; //Used to represent screen dimensions
     try
 {
System.out.println("Connecting to server .....");
socket = new Socket(ip, port);
System.out.println("Connection Established.");

          //Get default screen device
GraphicsEnvironmentgEnv=GraphicsEnvironment.getLocalGraphicsEnvironment();
GraphicsDevicegDev=gEnv.getDefaultScreenDevice();

          //Get screen dimensions
          Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
rectangle = new Rectangle(dim);

          //Prepare Robot object
robot = new Robot(gDev);

          //draw client gui
drawGUI();
          //ScreenSpyer sends screenshots of the client screen
newScreenSpyer(socket,robot,rectangle);
          //ServerDelegaterecievesserver commands and execute them
newServerDelegate(socket,robot);
       }
catch (UnknownHostException ex)
```

```
 {
ex.printStackTrace();
        }
 catch (IOException ex)
 {
ex.printStackTrace();
        }
catch (AWTException 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);
    }
}
```

2. Enum Commands

```
packageremoteclient;

/*
 * Used to represent commands which can be sent by the server
 */
publicenumEnumCommands {
    PRESS_MOUSE(-1),
    RELEASE_MOUSE(-2),
    PRESS_KEY(-3),
    RELEASE_KEY(-4),
    MOVE_MOUSE(-5);

privateint abbrev;

EnumCommands(int abbrev){
this.abbrev = abbrev;
    }

publicintgetAbbrev(){
return abbrev;
    }
}
```

3. ScreenSpyer

```
packageremoteclient;

importjava.awt.Rectangle;
importjava.awt.Robot;
importjava.awt.image.BufferedImage;
importjava.io.IOException;
importjava.io.ObjectOutputStream;
importjava.net.Socket;
importjavax.swing.ImageIcon;

/*
 * This class is responisble for sending sreenshotevery predefined duration
 */
classScreenSpyer extends Thread {

    Socket socket = null;
    Robot robot = null; // Used to capture screen
    Rectangle rectangle = null; //Used to represent screen dimensions
booleancontinueLoop = true; //Used to exit the program

publicScreenSpyer(Socket socket, Robot robot,Rectanglerect) {
this.socket = socket;
this.robot = robot;
rectangle = rect;
start();
    }

public void run(){
ObjectOutputStreamoos = null; //Used to write an object to the streem
```

```
try{
        //Prepare ObjectOutputStream
oos = new ObjectOutputStream(socket.getOutputStream());
        /*
         * Send screen size to the server in order to calculate correct mouse
         * location on the server's panel
         */
oos.writeObject(rectangle);
}
catch(IOException ex)
{
ex.printStackTrace();
      }

while(continueLoop){
        //Capture screen
BufferedImage image = robot.createScreenCapture(rectangle);
        /* I have to wrap BufferedImage with ImageIcon because BufferedImage class
         * does not implement Serializable interface
         */
ImageIconimageIcon = new ImageIcon(image);

        //Send captured screen to the server
try {
System.out.println("before sending image");
oos.writeObject(imageIcon);
oos.reset(); //Clear ObjectOutputStream cache
System.out.println("New screenshot sent");
        }
 catch (IOException ex)
{
ex.printStackTrace();
        }

        //wait for 100ms to reduce network traffic
```

```
try{
Thread.sleep(100);
}
catch(InterruptedException e)
{
e.printStackTrace();
            }
        }

    }

}
```

4. Server Delegate

```
packageremoteclient;

importjava.awt.Robot;
importjava.io.IOException;
importjava.net.Socket;
importjava.util.Scanner;

/*
 * Used to recieve server commands then execute them at the client side
 */
classServerDelegate extends Thread {

   Socket socket = null;
   Robot robot = null;
booleancontinueLoop = true;

publicServerDelegate(Socket socket, Robot robot) {
this.socket = socket;
this.robot = robot;
start(); //Start the thread and hence calling run method
   }

public void run(){
      Scanner scanner = null;
try {
         //prepare Scanner object
System.out.println("Preparing InputStream");
scanner = new Scanner(socket.getInputStream());

while(continueLoop){
            //recieve commands and respond accordingly
```

```
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();
        }
    }

}
```

Remote Server:

1. **Client Command Sender**

```
packageremoteserver;

importjava.awt.Rectangle;
importjava.awt.event.KeyEvent;
importjava.awt.event.KeyListener;
importjava.awt.event.MouseEvent;
importjava.awt.event.MouseListener;
importjava.awt.event.MouseMotionListener;
importjava.io.IOException;
importjava.io.PrintWriter;
importjava.net.Socket;
importjavax.swing.JPanel;

classClientCommandsSender implements KeyListener,
MouseMotionListener,MouseListener {

private Socket cSocket = null;
privateJPanelcPanel = null;
privatePrintWriter writer = null;
private Rectangle clientScreenDim = null;

ClientCommandsSender(Socket s, JPanel p, Rectangle r) {
cSocket = s;
cPanel = p;
clientScreenDim = r;
      //Associate event listners to the panel
cPanel.addKeyListener(this);
cPanel.addMouseListener(this);
```

```
cPanel.addMouseMotionListener(this);
try {
            //Prepare PrintWriter which will be used to send commands to
            //the client
writer = new PrintWriter(cSocket.getOutputStream());
        }
 catch (IOException ex)
{
ex.printStackTrace();
        }

    }

    //Not implemeted yet
public void mouseDragged(MouseEvent e) {
    }

public void mouseMoved(MouseEvent e) {
doublexScale = clientScreenDim.getWidth()/cPanel.getWidth();
System.out.println("xScale: " + xScale);
doubleyScale = clientScreenDim.getHeight()/cPanel.getHeight();
System.out.println("yScale: " + yScale);
System.out.println("Mouse Moved");
writer.println(EnumCommands.MOVE_MOUSE.getAbbrev());
writer.println((int)(e.getX() * xScale));
writer.println((int)(e.getY() * yScale));
writer.flush();
    }

    //this is not implemented
public void mouseClicked(MouseEvent e) {
    }

public void mousePressed(MouseEvent e) {
System.out.println("Mouse Pressed");
```

```
writer.println(EnumCommands.PRESS_MOUSE.getAbbrev());
int button = e.getButton();
intxButton = 16;
if (button == 3) {
xButton = 4;
        }
writer.println(xButton);
writer.flush();
    }

public void mouseReleased(MouseEvent e) {
System.out.println("Mouse Released");
writer.println(EnumCommands.RELEASE_MOUSE.getAbbrev());
int button = e.getButton();
intxButton = 16;
if (button == 3) {
xButton = 4;
        }
writer.println(xButton);
writer.flush();
    }

    //not implemented
public void mouseEntered(MouseEvent e) {
    }

    //not implemented
public void mouseExited(MouseEvent e) {

    }

    //not implemented
public void keyTyped(KeyEvent e) {
    }
```

```
public void keyPressed(KeyEvent e) {
System.out.println("Key Pressed");
writer.println(EnumCommands.PRESS_KEY.getAbbrev());
writer.println(e.getKeyCode());
writer.flush();
    }

public void keyReleased(KeyEvent e) {
System.out.println("Mouse Released");
writer.println(EnumCommands.RELEASE_KEY.getAbbrev());
writer.println(e.getKeyCode());
writer.flush();
    }

}
```

2. Client Handler

```
packageremoteserver;

importjava.awt.BorderLayout;
importjava.awt.Rectangle;
importjava.beans.PropertyVetoException;
importjava.io.IOException;
importjava.io.ObjectInputStream;
importjava.net.Socket;
importjavax.swing.JDesktopPane;
importjavax.swing.JInternalFrame;
importjavax.swing.JPanel;

classClientHandler extends Thread {

privateJDesktopPane desktop = null;
private Socket cSocket = null;
privateJInternalFrameinterFrame = new JInternalFrame("Client Screen",
true, true, true);
privateJPanelcPanel = new JPanel();

publicClientHandler(Socket cSocket, JDesktopPane desktop) {
this.cSocket = cSocket;
this.desktop = desktop;
start();
    }

    /*
     * Draw GUI per each connected client
     */
```

```
public void drawGUI(){
interFrame.setLayout(new BorderLayout());
interFrame.getContentPane().add(cPanel,BorderLayout.CENTER);
interFrame.setSize(100,100);
desktop.add(interFrame);
try {
        //Initially show the internal frame maximized
interFrame.setMaximum(true);
      }
catch (PropertyVetoException ex)
{
ex.printStackTrace();
      }
      //this allows to handleKeyListener events
cPanel.setFocusable(true);
interFrame.setVisible(true);
   }

public void run(){

      //used to represent client screen size
      Rectangle clientScreenDim = null;
      //Used to read screenshots and client screen dimension
ObjectInputStreamois = null;
      //start drawing GUI
drawGUI();

try{
        //Read client screen dimension
ois = new ObjectInputStream(cSocket.getInputStream());
clientScreenDim =(Rectangle) ois.readObject();
}catch(IOException ex){
ex.printStackTrace();
}catch(ClassNotFoundException ex){
ex.printStackTrace();
```

```
        }
        //Start recieveing screenshots
newClientScreenReciever(ois,cPanel);
        //Start sending events to the client
newClientCommandsSender(cSocket,cPanel,clientScreenDim);
    }

}
```

3. Client Screen Receiever

```
packageremoteserver;

importjava.awt.Graphics;
importjava.awt.Image;
importjava.io.IOException;
importjava.io.ObjectInputStream;
importjavax.swing.ImageIcon;
importjavax.swing.JPanel;

/**
* ClientScreenReciever is responsible for recieving client screenshot and displaying
 * it in the server. Each connected client has a separate object of this class
 */
classClientScreenReciever extends Thread {

privateObjectInputStreamcObjectInputStream = null;
privateJPanelcPanel = null;
privatebooleancontinueLoop = true;

publicClientScreenReciever(ObjectInputStreamois, JPanel p) {
cObjectInputStream = ois;
cPanel = p;
      //start the thread and thus call the run method
start();
   }

public void run(){

try {

            //Read screenshots of the client then draw them
```

```
while(continueLoop){
                //Recieve client screenshot and resize it to the current panel size
ImageIconimageIcon = (ImageIcon) cObjectInputStream.readObject();
System.out.println("New image recieved");
                Image image = imageIcon.getImage();
image = image.getScaledInstance(cPanel.getWidth(),cPanel.getHeight()
,Image.SCALE_FAST);
                //Draw the recieved screenshot
                Graphics graphics = cPanel.getGraphics();
graphics.drawImage(image, 0, 0, cPanel.getWidth(),cPanel.getHeight(),cPanel);
              }
           }
 catch (IOException ex)
{
ex.printStackTrace();
         }
 catch(ClassNotFoundException ex)
{
ex.printStackTrace();
         }
     }
}
```

4. Enum Commands

```
packageremoteserver;

/**
 * Used to represent commands which can be sent by the server
 */
publicenumEnumCommands {
    PRESS_MOUSE(-1),
    RELEASE_MOUSE(-2),
    PRESS_KEY(-3),
    RELEASE_KEY(-4),
    MOVE_MOUSE(-5);

privateint abbrev;

EnumCommands(int abbrev){
this.abbrev = abbrev;
    }

publicintgetAbbrev(){
return abbrev;
    }
}
```

5. Server Initiator

```
packageremoteserver;

importjava.awt.BorderLayout;
importjava.io.IOException;
importjava.net.ServerSocket;
importjava.net.Socket;
importjavax.swing.JDesktopPane;
importjavax.swing.JFrame;
importjavax.swing.JOptionPane;

/*
 * This is the entry class of the server
 */
public class ServerInitiator {
   //Main server frame
privateJFrame frame = new JFrame();
   //JDesktopPane represents the main container that will contain all
   //connected clients' screens
privateJDesktopPane desktop = new JDesktopPane();

public static void main(String args[]){
       String port = JOptionPane.showInputDialog("Please enter listening port");
newServerInitiator().initialize(Integer.parseInt(port));
   }

public void initialize(int port){

try {
ServerSocketsc = new ServerSocket(port);
          //Show Server GUI
drawGUI();
          //Listen to server port and accept clients connections
```

```
while(true){
            Socket client = sc.accept();
System.out.println("New client Connected to the server");
            //Per each client create a ClientHandler
newClientHandler(client,desktop);
          }
       }
catch (IOException ex)
 {
ex.printStackTrace();
       }
    }

    /*
     * Draws the main server GUI
     */
public void drawGUI(){
frame.add(desktop,BorderLayout.CENTER);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          //Show the frame in a maximized state
frame.setExtendedState(frame.getExtendedState()|JFrame.MAXIMIZED_BOTH);
frame.setVisible(true);
    }
}
```

TESTING

Unit Testing:

Unit Testing basically involves the individual testing of various modules of program together with associated data, using procedures to analyse whether they are fit for use or not.

Each test case is different from the other ones. With the help of unit testing complexity of testing is managed. It facilitates debugging and encourages parallel testing. Modules are combined using two approaches:

(i) Non- incremental – all the modules are tested independently and then all the modules are combined and whole program is tested.
(ii) Incremental – in incremental each module is added to the tested collection or we can say step wise retesting takes place.

Firstly we tested all the modules on client side to see whether there was any syntactical error in any of the modules to be run at client side.
Secondly, all the modules of server side are tested individually for syntactical errors.

Integration testing:

It is a phase in software testing in which various software modules are combined and tested as a group. It takes place after unit testing and before validation testing. The modules that have been unit tested are inputted in integration testing, are then aggregated in larger groups and applies tests defined in an integrated test plan to those aggregates.

The purpose of integration testing is to fulfill functional, performance and reliability requirements. Some different types of integration testing are Big Bang, Top Down and Bottom up.

Big Bang Testing: In big bang approach all the developed modules are aggregated to form a complete software system and then used for integration testing. This method is very effective in saving time. However if test cases and their results are not recorded properly, the whole integration

process will be highly complicated.

Bottom Up Testing: Bottom up testing is an approach to integrated testing where the lower level components are tested first and then moving on to the testing of higher level components and the process is repeated until the component at the top level of hierarchy is tested. This approach is very helpful only when all the modules of same development level are ready.

Top Down Testing: It is an approach to integrated testing where the top level modules are tested first and the lower modules are tested step by step until the end of the related module.

Sandwich testing is an approach in which top down and bottom up testing are combined.

In our project integration testing is being performed by the use of **Bing Bang** approach.

All the modules were combined initially together both at client and server side. At the client side errors regarding the sending of screen co-ordinates came initially. The co-ordinates did not map well at the server side.

At Server side co-ordinates were passed but screen view could not be received properly. Also the key strokes were getting passed on to the client side but mouse movement was producing the time lag.

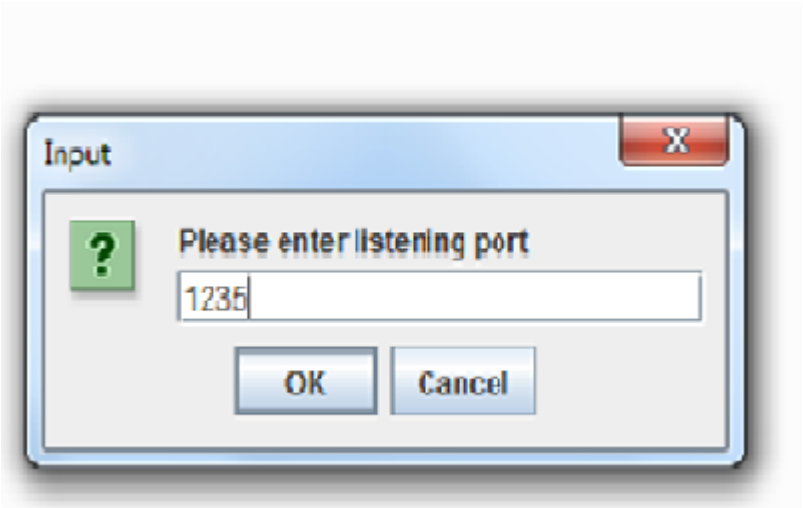
Verification and validation Testing: It is a process in which a software system meets specifications and then fulfills the intended purpose. It is also known as software quality control. Validation checks the product design i.e. the software meets the user requirements or not. It is the process of evaluating software during or at the end of development process to determine whether it satisfies the specified requirements.

Verification testing is the process of evaluating software to determine whether the products satisfy the condition imposed at the start f that phase. Both verification and validation are related to the concepts of quality and software quality assurance.

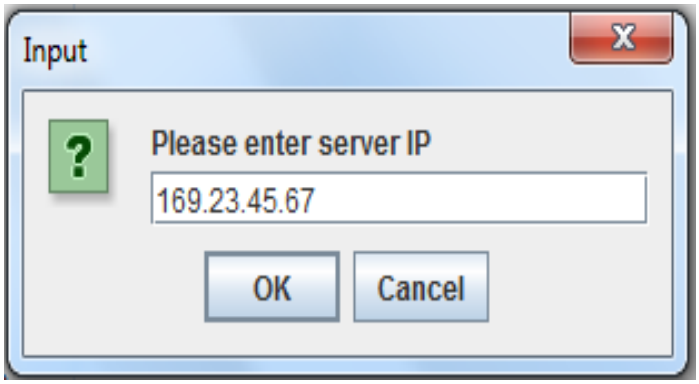
In this method we compared the various softwares in market and tried to analyse the quality of software made by us. We checked upon the various requirements of user to be met. Since ours is a diagnostic tool to be used a source for desktop monitoring, emphasis was on to remove all the discrepancies in our product and make the software as much user friendly as it can be made.

SYSTEM IMPLEMENTATION

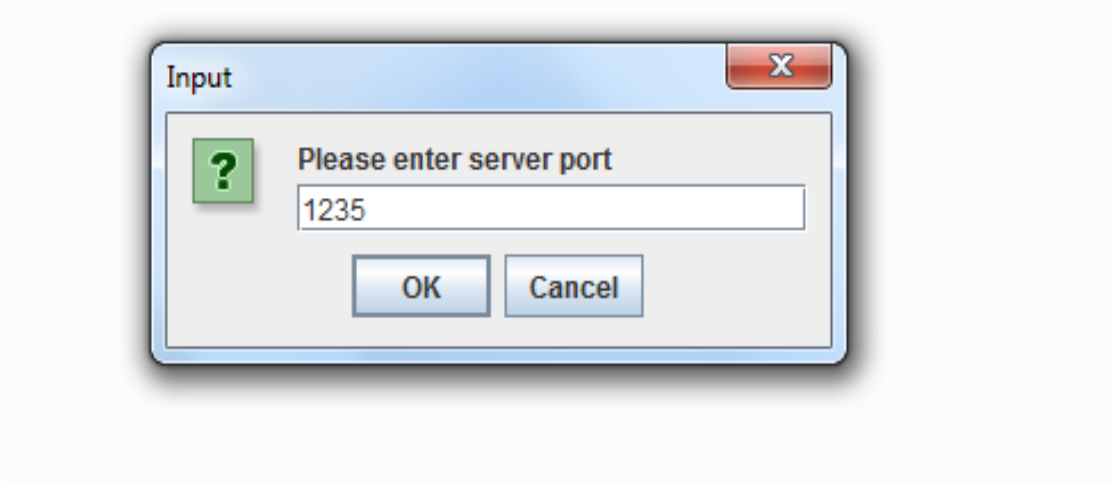
Server Side Authentication



Client Side Authentication

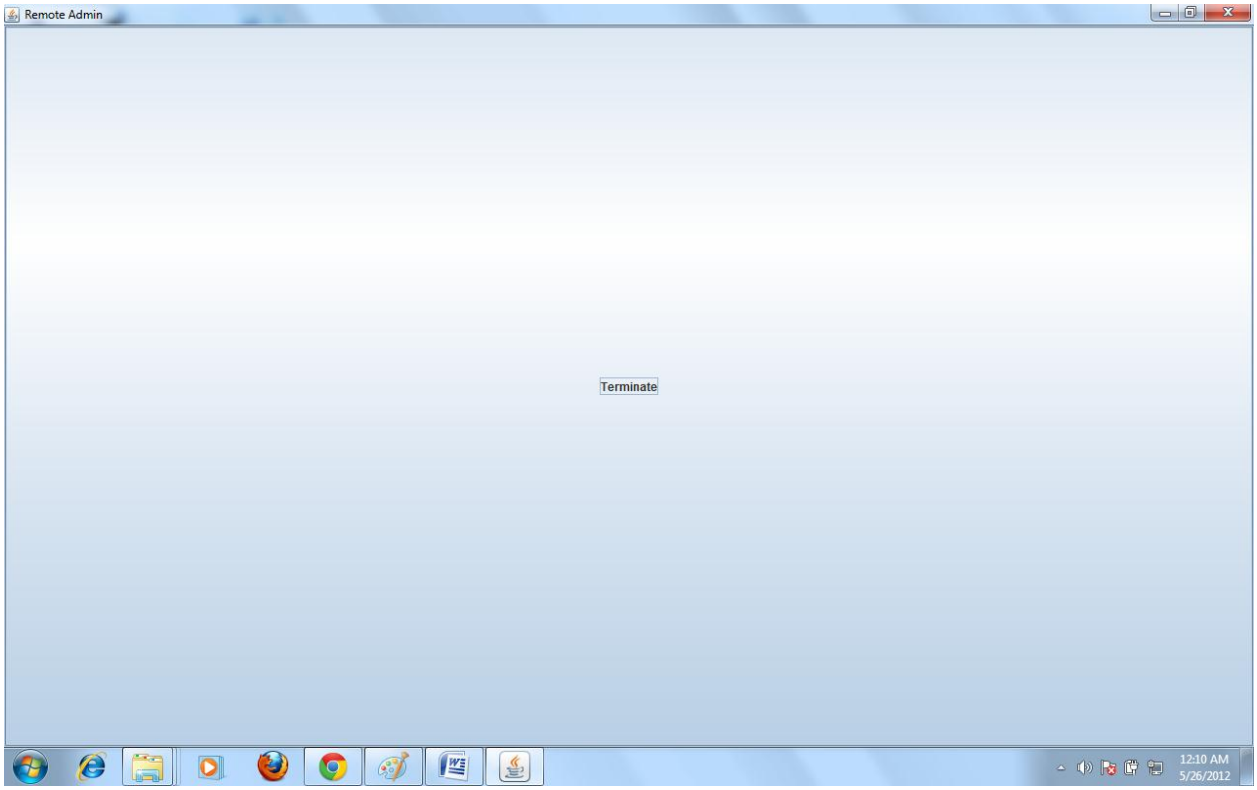


Port Number Should Be Same On Both Client And Server Side

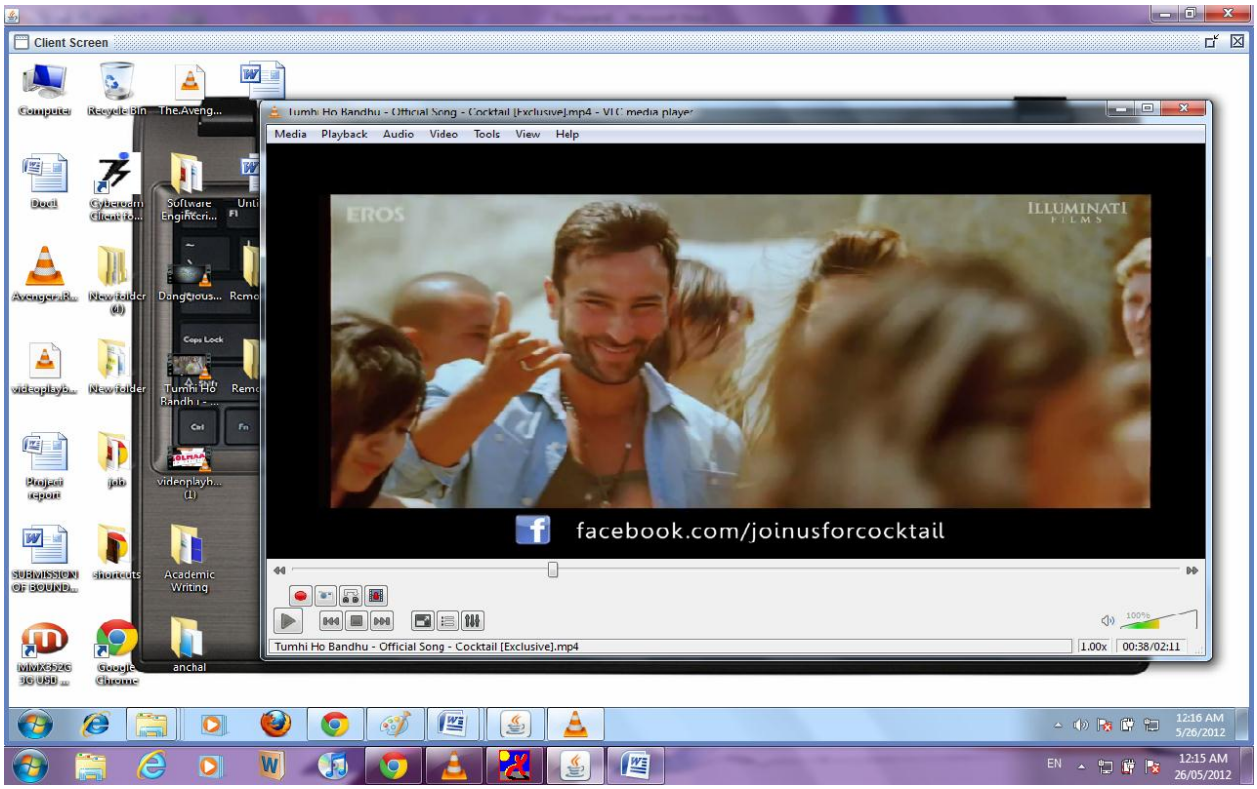


Server captures the screen of client.

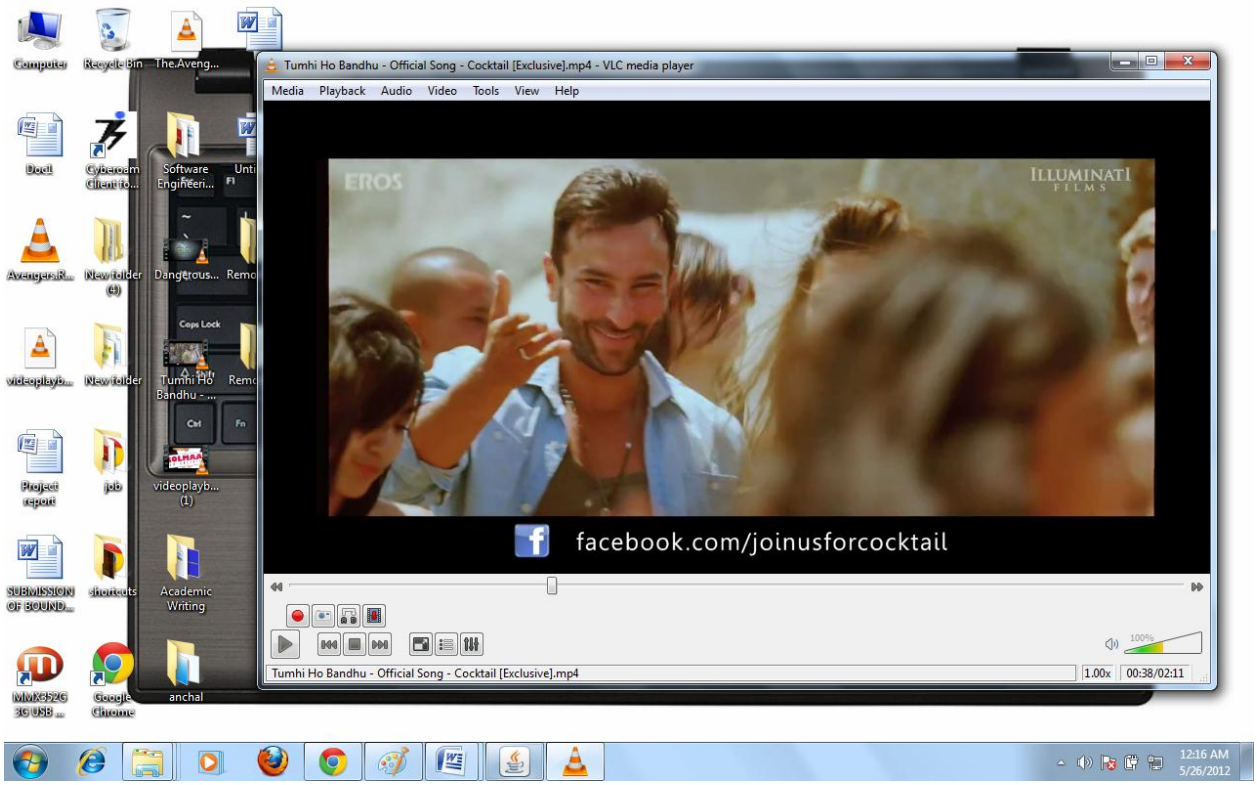
Connection is established with the server at the client side.



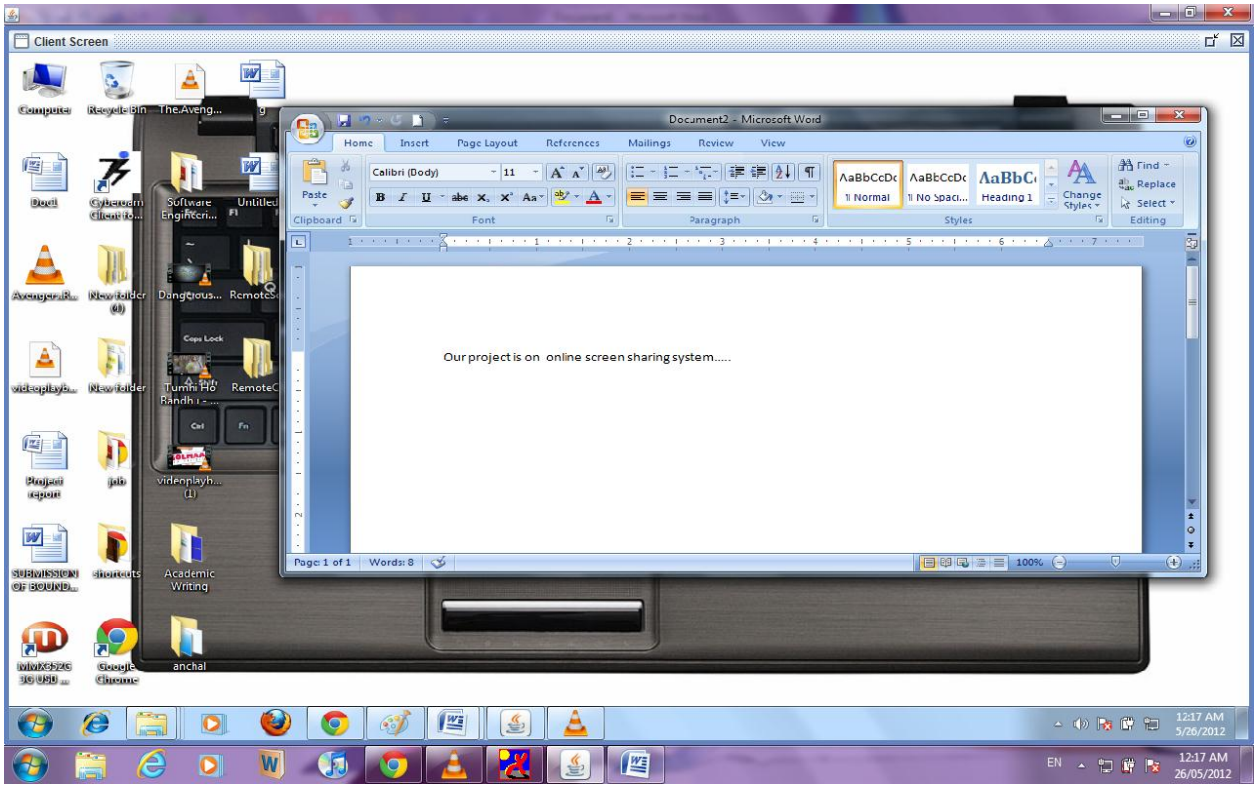
Video played at client side is captured and viewed at server side.



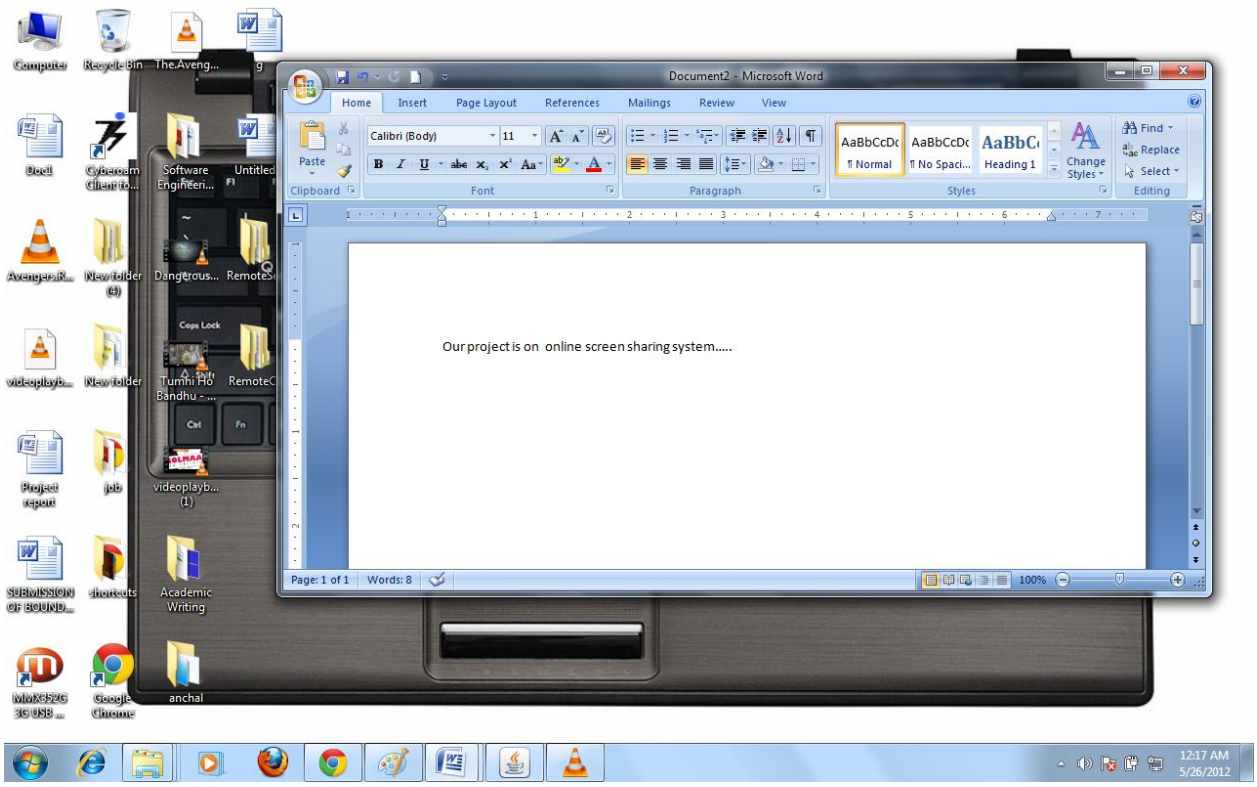
Video played at client side to be shared at server side.



Document opened at client side, viewed at server side and content is being typed from server side.



Document opened at client side while the content is entered from server side.



CONCLUSION

At last we can conclude that after completing this project we have learned how to work with various java technologies like AWT, Swing and Robot Class. We also learnt how to work under extreme pressure conditions and delivering high performance.

Our overall learning in this project has been splendid. We all learnt how to work together in a team in spite of all the odds which came in our path.

We faced various difficulties while implementing this project, most of which were overcome by the help of our project guide Dr. Ravi Rastogi.

Also we have some proposed future work for this project like the video lag needs to be minimized much more and also the audio transfer feature is to be included so that the sound played at client side could be heard at the server side.

References

1) Comparison of Java Remote Desktop Projects – Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Comparison_of_Java_Remote_Desktop_projects accessed on 10th ferbuary 2012.
2) Remote desktop software – Wikipedia the free encyclopedia, http://en.wikipedia.org/wiki/Remote_desktop_software accesses on 12th november 2011.
3) The Software Development Life Cycle, 2000-2005 Digital Publications LLC
4) A Tutorial on Socket Programming in Java, Natarajan Meghanathan, Jackson State University.
5) Software Testing Techniques, Lu Luo, Institute For Software Research International, Carnegie Mellon University
6) Advanced Java Programming Swing,Eran Werner, Tel-Aviv University Summer,2005
7) Java 2 Black Book, Steven Holzener et al., Paraglyph Press