INDEX

ABSTARCT

Today computers have become an integral part of everyone's life. Use of computers is not only restricted for corporate use but also for personal use and intercommunication purpose. For all these different purposes networking or network has become the magic word. Today, lots of information and resources are shared all over the world through the network. The buzzword or the revolutionary term originated from this is global network i.e. Internet. But along with all the advancements and developments, the danger of misusing the network has also increased.

The main objective behind development of "Remote Desktop Monitor" to build a server application that allows to view or even control the desktop session on another machine that is running a compatible client application, with the efficiency to provide the real time information about the remote machine. Remote access allows users with remote computers to create a logical connection to an organization network or the Internet. Microsoft operating systems include remote access clients and remote access servers that support both dial-up and virtual private network connections. Microsoft operating systems include a rich set of remote access and security features that provide flexibility and security for your remote access solution.

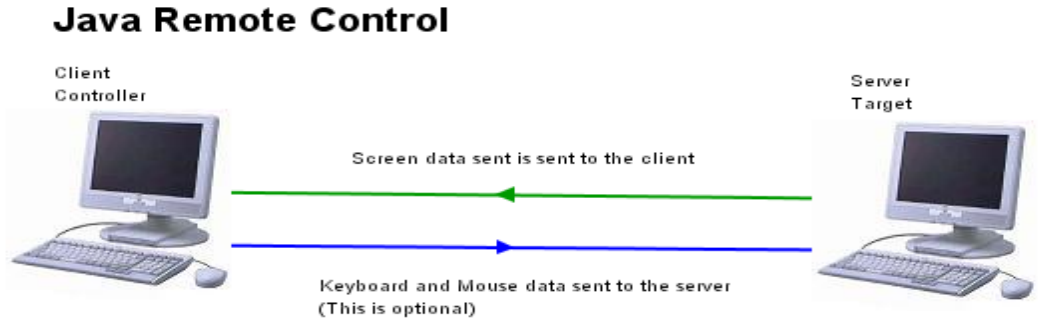
INTRODUCTION

This project is a network based one. The primary objective behind development of “Remote desktop Application” is to provide a mechanism by which system administrators could remotely connect to a desktop machine and resolve basic problems for users. However, the umbrella term "Remote Desktop" encompasses much more functionality and use cases. The project aims at developing an application to access and interact with a remote desktop. Aim is to remotely administer a system over the network. It may comprise of following tasks: -

1 View a desktop remotely.
2 Interact with remote desktop.
3 Open a connection to user desktop, having user aware of the connection.
4 Open a connection to user desktop, without having user aware of the connection.

1.1 System Overview

The Remote desktop Monitor will allow the system administrator to use or to remotely control the other systems through a simple application. This software intends to leverage existing technologies and to utilize them in a new way. The proposed Remote desktop will exist in a distributed computer environment such as the Internet but will also be accessible from within an intranet such as an office or in the home itself. The proposed system can best be illustrated with the following diagram



1.2 Objective

Remote desktop Application is a program that allows the user to connect to a more computer. See that computer's desktop and interact with it as if it were local. People use remote desktop capability to do a variety of things remotely, including the following:

- Access a home computer from other locations.
- Fix a computer problem.
- Perform administrative tasks.
- Remote desktop connectivity relies upon any of a number of protocols
- Remote desktop software exists for most operating systems and platforms, including handheld computing devices

1.3 SCOPE

This application has a very wide area of usage. As the number of users increase the scope of the project goes on. There is no limit for this application this can be used to any type of page by anyone based on the restrictions given by the developer. This project is useful in lab monitoring i.e. what a student is doing in present It is, in spirit, a remote display system which allows you to view a computing 'desktop' environment not only on the machine where it is management, but from anywhere on the Internet and as of a wide variety of machine architectures. With this type of application the strength of a page can be increased. The knowledge can be distributed to a large extent.

2 PROBLEM STATEMENTS

Existing System:-

Ample options are there in now a day s operating systems it to execute applications at the remote end. The basic services used by these operating systems today promote executions of the applications at the remote end with just restricted access. The current existing systems are potentially good systems, which do allow us to remotely connect to the remote machines and access their respective desktops. But most of the existing systems are quite difficult to use and implement for a layman.

Proposed System:-

PROBLEMS WITH THE EXISTING SYSTEM Administrator is not having full control There is no provision to reboot or shutdown Supports only one remote command on the remote machine at the same time Never gets the feeling that we are using the remote machine We cannot capture the remote systems Desktop. Network Administrator Tool provides remote service to its entire client over the network. It acts as a network administrator to its client to provide remote service like remote shutdown, remote logoff, remote file transfer, and remote desktop sharing and remote chatting. If the Remote operations such as remote shutdown, remote logoff requests arrive from client then it parses the request and provides service to its corresponding client. File transfer operation from server to the requested client:- First receives the request and recognize the file name, if the file exists in server transfers it otherwise sends a file not found error message.

Remote Desktop Sharing enables the administrators to gain access to remote Windows desktops in the network. It is a tool enabling access from anywhere in the network without requiring any native client. It allows almost all operations to be

performed on the remote desktop. Remote Desktop Sharing lets users call a remote computer to access its shared desktop and applications. With the desktop Sharing we can operate our office computer from home or vice versa. Remote Messaging is a small application that facilitates communication between different hosts on the same local area network. It does not require a central server and uses very little bandwidth by taking advantage of a lightweight protocol and UDP packets.

3 PROJECT FEASIBILITY STUDY

All projects are feasible, given unlimited resources and infinite time. But the development of software is plagued by the scarcity of resources and difficult delivery rates. It is prudent to evaluate the feasibility of the project at the earliest possible time.

Three key considerations are involved in feasibility analysis.

Technical Feasibility

The REMOTE VIRTUAL DESKTOP software is developed using the java environment. The reason for using java, as the development platform is that, java is an Object Oriented Language which handles most of the networking concepts. Since java is a platform independent language, the class files can be executed on any operating system easily.

Economic Feasibility

This is the most frequently used method for evaluating the effectiveness of a system. It is also called as a cost analysis. The **Remote Virtual Desktop** project, which is used to control all the remote systems in a network, requires resources

such as the software and hardware components that support the Remote Control through java project effectively. Since all the clients are usually connected to the server in any organization, it reduces the cost factor. Hence there is no need of further physical connection to be established between the server and the client.

Operational Feasibility

The Remote Virtual Desktop project is a user-friendly tool developed in order to make the operations of the administrator much better. It will be easy for the administrator to handle all the systems in the network from the server itself which helps in increasing the operational efficiency of the administrator.

3.1 TECHNICAL ASPECTS

3.1.1 H/W SPECIFICATION-

1. 512MB of RAM
2. PIV processor(1.3GHz) recommended
3. Minimum 80 GB capacity hard disk
4. LAN CABLE
5. SWITCH
6. NIC CARD

3.1.2 S/W SPECIFICATION-

1. Operating System:Windows Xp, Win7, Win8.1 etc.
2. Tools: Java

3.2 ECONOMICAL ASPECTS

The project developed, Consumer Complaint System was within budget and producing the desired results. The labor or the human ware consisted of the two group members of our project. The output consisted of getting the desired results. Thus with the consideration of the inputs, the outputs were achieved successfully. The project was within limit. The inputs didn't overdo the outputs.

Economical aspects aim to determine the impact of the system on the users. The system developing has an influence on its users. Our system "Consumer Complaint System" was new for them but it was simple enough for any naïve person to understand. The evolution of this new system required no special training for the users. Consumer Complaint System was found to be feasible in this regard. The system developed would be user friendly and no complexities would be involved in its functionalities

4. EXISITING SYSTEM

There are many existing systems which work as a connection mediator between Server and the client desktop. One of the applications is used for controlling Client Machine. Another application is based on Central Server, in which the server and client are connected to the central server. Here the application logic is executed on the central server. So the speed of operation is low and the whole system solely depends on the central server. Network Administrator is a crucial job to monitor and manage systems in a LAN.

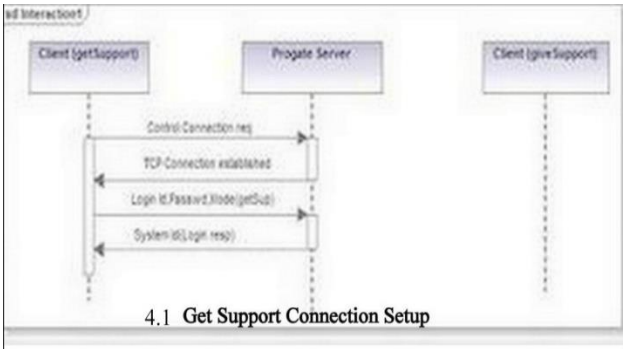
LAN has different platforms and resources so if the administrator want to monitor or share the resources on the remote systems in the LAN, operating systems doesn't have compatibility

5. PROCESS DESCRIPTION/METHODOLOGY

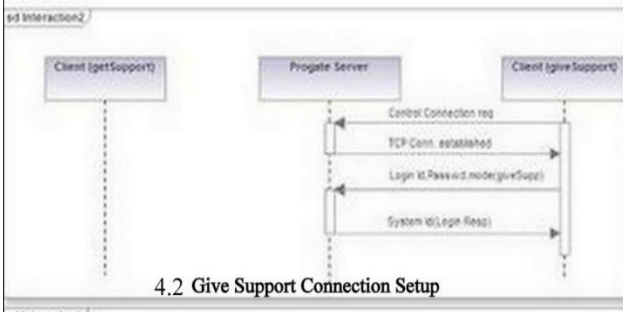
For the tool to be developed, a very simple methodology is used in which the server will be running in listening mode [8]. As soon as the client logins, an executable will run on the client side which will contain the VNC server and the VNC client, these are responsible for desktop sharing as well as desktop accessing. This executable is responsible for connecting to the server and doing further work. The shared desktop and the sharing desktop are connected to the server via a control

connection. This will create an entry in the control connection table which is present at the server location. Some of the columns in data connection table will be filled when the sharing desktop is connected and the entry will be full if the accessing desktop completes all the credentials and asks for the permission to access the desktop.

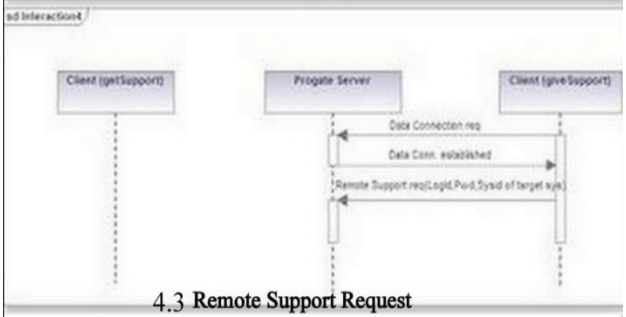
The Client interacts with the Server through the following functions: Get Support Connection Setup method is called when a client asks to connect to Server and makes a Control Connection, Figure 1 Give Support Connection Setup provides Control Connection to the client, Figure 2To request for data transmission Remote Support Request is done, Figure 3 Accept Support Acknowledgement, accepts the data connection request and a Data Connection gets established, Figure 4 After Data Connection establishment Data Exchange, Figure 4.5, occurs synchronously among the clients joined during a session. This way the server interacts with the clients who are connecting to it, and the data connections and data transfer are facilitated by the connection handler method which manages the file descriptors.



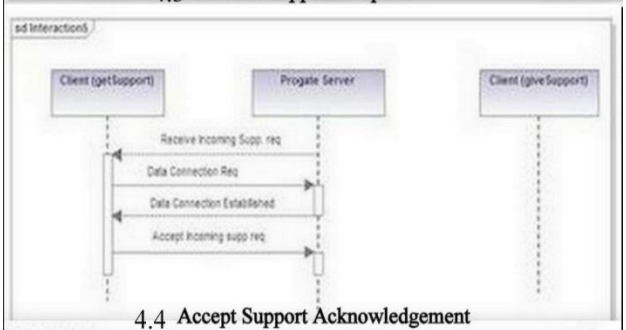
4.1 Get Support Connection Setup



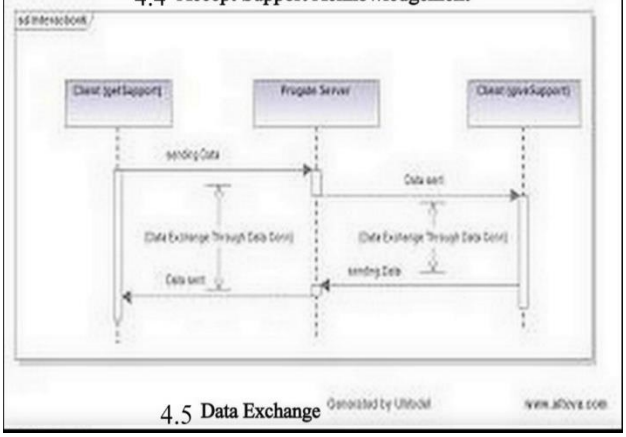
4.2 Give Support Connection Setup



4.3 Remote Support Request



4.4 Accept Support Acknowledgement



4.5 Data Exchange

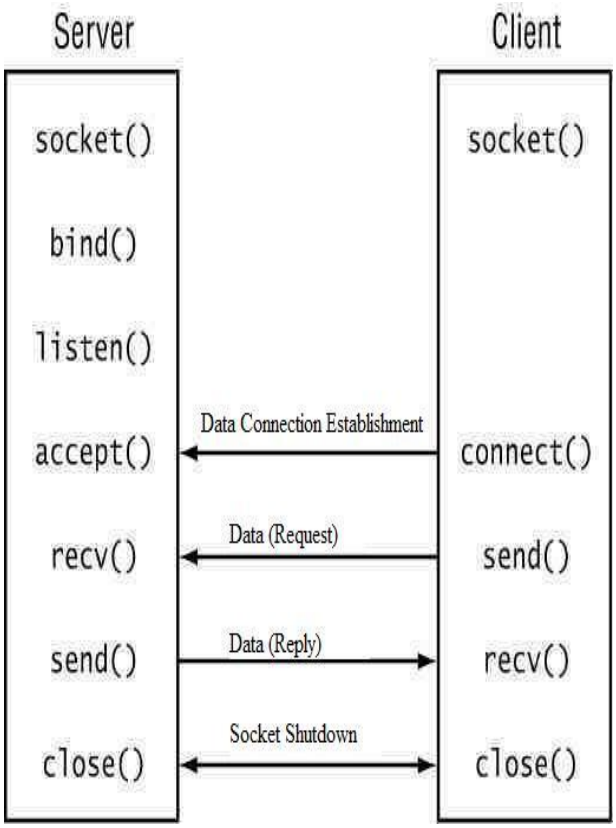


Figure . Server Client Communication

Working

Desktop sharing [8] such that one could access it from a distant place is kept simple. As soon as user registers using our web page an executable file is downloaded at user's machine and irrespective of whether it is host machine or guest machine. This executable file is downloaded at user's machine without any prompt to the user and since the size of file is too less, that is 52 kb, it is downloaded quickly. This downloaded file is stored in a temporary file (for example %temp% folder in windows) and hence it is executed. Now, the problem which may arise here is that the browser doesn't have the rights to execute the executable file, but the java applet could execute it. As the browser can run the java applet. So now we were able to download and execute the executable file at user's end without even bringing it in notice of user. Now, once this executable file is downloaded and executed it will start communicating with our Progate Server. And now the Server will just act as a bridge between the host client and the guest client and the executable file will run at both the clients.

Role of Server

The server acts as a reflector, Figure 3, present in the public domain and maintains Control Connection table and Data connection Table. Control Connections are made first at the time of user log-in and registration in the Progate. Once the entries in control tables have been made, the user is free to either host a new session or to join an existing session by providing the id of the host to which it wishes to connect to. Data Connections are made in addition to handle the data transfer between the stream sockets for a particular session. The session ends as the client logs out or the meeting is no longer continued by the host.

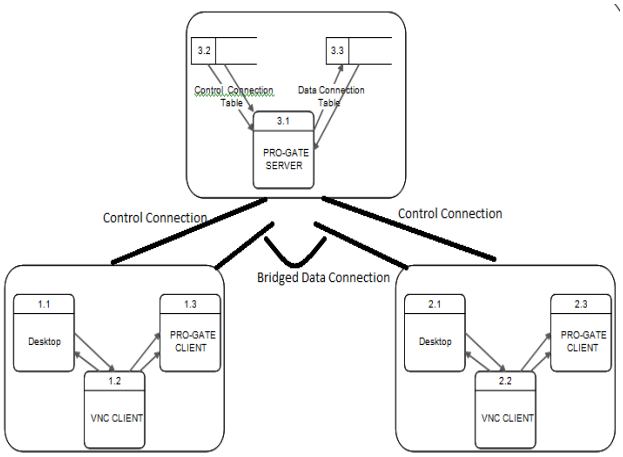


Figure . Server as a Reflector

Module Description

Remote Access Methodology:

We can access a client system through a server for which we will use java networking.

Searching for Active User:

We will search the whole network for the system running our client application par.

Getting access to all available user:

Get connected to the users who are running client application part to perform the desire task.

Sharing Views

In this module the server is shared throughout the active users

Sharing File

File can be shared throughout the active users

ROLE OF MOUDLE:

1. **Remote control**

   Remote control module provides remote operation such as remote shutdown, and remote logoff if request arrives from client. First it parses the request and provides service to its corresponding client. Remote controlis an electronic device used for the remote operation of a machine.

2. Remote File transfer

Remote file transfer module provides file transfer operation from server to the requested client. First receives the request and recognize the file name, if the file exists in server transfers it otherwise sends a file not found error message. *Remote File Transfer* contains information for transferring files to the remote location. Remote file transfers are used to transfer files from the development environment to a server that can be installed locally or remotely. We can transfer files between two connected PCs and chat with the remote user. Transfer files and folders between local and remotely accessed computers.

3. Remote Desktop Sharing

Remote Desktop Sharing module provides the administrators to gain access to remote Windows desktops in the network. It is a tool enabling access from anywhere in the network without requiring any native client. Administrator can directly access the remote system by sharing the requested system desktop. Desktop Sharing is a server application that allows to share current session with a user on another machine, who can use a client to view or even control the desktop. Desktop Sharing lets users call a remote computer to access its shared desktop and applications. With the Desktop sharing we can operate our office computer from our home or vice versa.

4. Remote Messaging

Remote Messaging is a small application that facilitates communication between different hosts on the same local area network. It does not require a central server and uses very little bandwidth by taking advantage of a lightweight protocol and UDP packets. Administrator can communicate with the remote

systems that are connected with in the local network administrator can communicate publicly or privately. Messaging is nothing but passing data to and from applications over the network which makes the synchronization of data simple. Messaging allows users across the network to exchange data in real time.

JAVA

Introduction

Java is an object oriented, multi thread programming language developed by Sun Microsystems in 1991. It is designed to be small, simple and portable across different platforms as well as operating systems. The popularity of Java is due to its unique technology that is designed on the basis of three key elements. They are the usage of applets, powerful programming language constructs and a rich set of significant object classes. The editor (i.e., where the programs are being written) can be Notepad, WordPad, MS-DOS editor, etc…). This provides system input and output capabilities and other utility functions in addition to classes that support networking, common Internet protocols and user interface toolkit functions.

Why Java is selected?

Java was designed to meet all the real world requirements with its key features, which are explained in the following paragraphs:

Simple and powerful.

Java was designed to be easy for the professional programmers to learn and use efficiently. Java makes itself simple by not having surprising features. Since it exposes the internal working of the machine, the programmers can perform his desired action without fear. Unlike other programming systems that provide dozens of complicated ways to perform a simple task, Java provides a small number of clear ways to achieve a given task. Today everyone is worried about safety and security. Using Java Compatible Browser, anyone can safety download applets without fear of viral infections or malicious intent. Java achieves this protection by confining a Java

program to Java execution environment and by making it inaccessible to other parts of the computer.

Portable.

In Java, the same mechanism that gives security helps in portability. Many types of computers and operating systems are used throughout the world and are connected to the Internet. For downloading programs through different platforms connected o the Internet, some portable, executable ode is needed. Java's answer to these problems is its well-designed architecture.

Object-oriented.

Java was not designed to be source code compatible with any other language. Java team gave a clean, usable, realistic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects. Most programs in use nowadays fail of the two reasons: memory management or exceptional conditions. Java forces the user to find mistakes in the early stages of the program development. At the same time, Java frees the user from having to worry about the most common causes of the programming errors. Java virtually rectifies the problem of memory management by managing memory allocation and automatic memory reallocation by providing garbage collection for unused objects.

Multithreaded.

Java was designed to meet the real-world requirements of creating interactive, networked programs. To achieve this, Java supports multithreaded programming, which allows user to write programs that perform many function simultaneously. The Java run-time system enables the user to construct smoothly running interactive systems. Java's easy-to-use approach to multithreading allows the user to think about the specific behavior of his own program, not the multitasking subsystem. The Java designers worked hard in attaining their goal "write once; run anywhere, anytime, forever" and as a result the Java Virtual Machine was developed. A main issue for the designers was that of code longevity and portability. One of the main problems is the execution speed of the program. Since Java is architecture-neutral it generates byte code that resembles machine code and are not specific to any processor.

Java Socket Programming

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

Important methods

Method	Description
1) public InputStream getInputStream()	returns the InputStream attached with this socket.
2) public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
3) public synchronized void close()	closes this socket

The java.net package provides support for the two common network protocols:

- **TCP:** TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
- **UDP:** UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

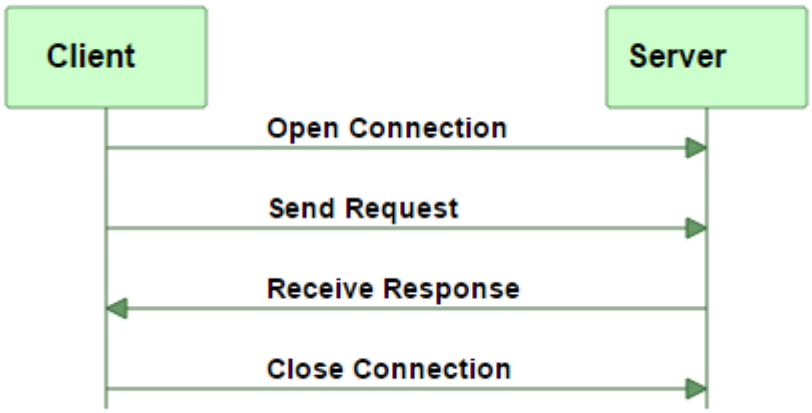
This tutorial gives good understanding on the following two subjects:

- **Socket Programming**: This is most widely used concept in Networking and it has been explained in very detail.
- **URL Processing**: This would be covered separately. Click here to learn about URL Processing in Java language.

The following steps occur when establishing a TCP connection between two computers using sockets:

- The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
- The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and port number. If communication is established, the client now has a Socket object capable of communicating 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 socket.

- **Java TCP Networking Basics**
- Typically a client opens a TCP/IP connection to a server. The client then starts to communicate with the server. When the client is finished it closes the connection again. Here is an illustration of that:



A client may send more than one request through an open connection. In fact, a client can send as much data as the server is ready to receive. The server can also close the connection if it wants to.

Java Socket's and ServerSocket's

When a client wants to open a TCP/IP connection to a server, it does so using a **Java Socket**. The socket is told what IP address and TCP port to connect to and the rest is done by Java.

If you want to start a server that listens for incoming connections from clients on some TCP port, you have to use a **Java ServerSocket**. When a client connects via a client socket to a server's ServerSocket, a Socket is assigned on the server to that connection. The client and server now communicates Socket-to-Socket.

Socket's and ServerSocket's are covered in more detail in later texts.

Internet Addresses

Internet addresses are manipulated in Java by the use of the InetAddress class. InetAddress takes care of the Domain Name System (DNS) look-up and reverse look-up; IP addresses can be specified by either the host name or the raw IP address. InetAddress provides methods to getByName(), getAllByName(), getLocalHost(),

getAddress(), etc. IP Addresses IP addresses are a 32-bit number, often represented as a "quad" of four 8-bit numbers separated by periods. They are organized into classes (A, B, C, D, and E) which are used to group varying numbers of hosts in a hierarchical numbering scheme.

Class A:- 1.0.0.0 to 126.255.255.255, inclusive. About 16 million IP addresses in a class A domain.

Class B:- 128.1.0.0 to 191.254.255.255, inclusive. About 64 thousand IP addresses in a class B domain**.**

Class C:- 192.0.1.0 to 223.255.254.255, inclusive. 256 IP addresses in a class C domain.

Class D:- 224.0.0.1 to 239.255.255.255, inclusive, denote multicast groups

Class E:- 240.0.0.0 to 254.255.255.255, inclusive. Reserved for future use. The IP address 127.0.0.1 is special, and is reserved to represent the loopback or "localhost" address.

Client/Server Computing

You can use the Java language to communicate with remote file systems using a client/server model. A server listens for connection requests from clients across the network or even from the same machine. Clients know how to connect to the server via an IP address and port number. Upon connection, the server reads the request sent by the client and responds appropriately. In this way, applications can be broken down into specific tasks that are accomplished in separate locations. The data that is sent back and forth over a socket can be anything you like. Normally, the client sends a request for information or processing to the server, which performs a task or sends data back. You could, for example, place an SQL shell on the server and let people talk to it via a simple client "chat" program

Port number

The port number field of an IP packet is specified as a 16-bit unsigned integer. This means that valid port numbers range from 1 through 65535. (Port number 0 is reserved and can't be used). Java does not have any unsigned data types; Java's short data type is 16 bits, but its range is -32768 to 32767 since it is a signed type. Thus, short is not large enough to hold a port number, so all classes which use or

return a port number must represent the port number as an int. In the JDK 1.1, using an int with a value greater than 65535 will generate an IllegalArgumentException. In the JDK 1.0.2 and earlier, values greater than 65535 are truncated and only the loworder 16 bits are used. Port numbers 1 through 255 are reserved by IP for well-known services. A well known service is a service that is widely implemented which resides at a published, "well-known", port. If you connect to port 80 of a host, for instance, you may expect to find an HTTP server. On UNIX machines, ports less than 1024 are privileged and can only be bound by the root user. This is so an arbitrary user on a multi-user system can't impersonate well-known services like TELNET (port 23), creating a security problem. Windows has no such restrictions, but you should program as if it did so that your applications will work cross-platform.

FLOW CHARTS

Before solving a problem with the help of a computer, it is essential to plan the solution in a step-by-step manner. Such a planning is represented symbolically with the help of flow chart. It is an important tool of system analysts and Programmers for tracing the information flow and the logical sequence in data processing Logic is the essence of a flow chart. A flow chart is the symbolic representation of step-by-step solution of a given problem, and it indicates flow of entire process, the sequence of the data input, operations, computations, decisions, results and other relevant information. Pertaining to a particular problem, a flow chart helps us in the complete understanding of the logical structure of a complicated problem and in documenting the method used. It would be seen that the flow chart is a very convenient method of organizing the logical steps and deciding what, when and how to proceed with various processes. The logic should be depicted in the flow charts. Computerization of the data without a flow chart is like constructing the building without a proper design and detailed drawings.

Kinds of the flow charts

1. System Flow Chart:-

The system analyst to describe data flow and operations for the data processing cycle uses these. A system flow chart defines the broad processing in the organizations, showing the origin of the data, filling structure, processing to be performed, output that is to generate and necessity of the offline operation.

2. Program Flow Chart (or) Computer Procedure flow chart:-

The programmers to describe the sequence of operations and the decision of a particular problem normally use these. A program flow chart plans the program structure and also serves the purpose of documentation for a program, while chart is to be retained and used at a later date either by the original programmer or others.

Advantages:

Apart from, the DFDS the flow charts has been helping the programmer to develop the programming logic and to serve as the documentation for a Completed program, it has the following advantages:-

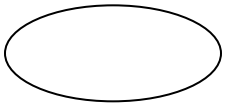
1. They help for the easy understanding of the logic of a process or a procedure
2. It is a better communicating tool than writing in Words.
3. It is easy to find the conditions, which are responsible for the actions.
4. It is an important tool for planning and designing the new system.
5. It clearly indicates the role-played at each level.
6. It provides an overview of the system and also demonstrates the relationship between the various steps.
7. It facilitates troubleshooting.
8. It promotes logical accuracy.

Disadvantages:

1. Communication lines are not always easy to show.
2. The charts are sometimes complicated.
3. Reproduction is difficult.
4. They are hard to modify.

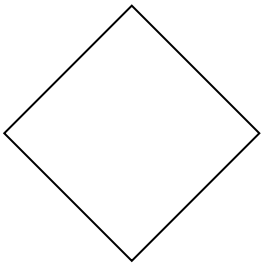
Symbols used in the flow charts:-

TERMINAL POINT (START/STOP)

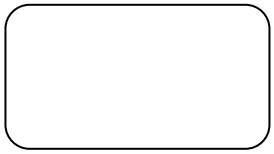


DECISION BOX

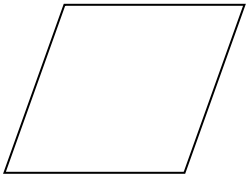
PROCESS



ALTERNATE PROCESS



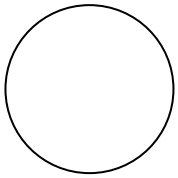
DATA



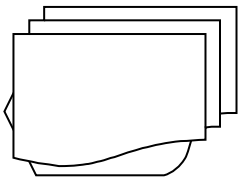
PRE DEFINED PROCESS

DOCUMENT (PRINT OUT)

PROCESS



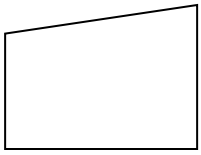
MULTIDOCUMENCONNECTOR



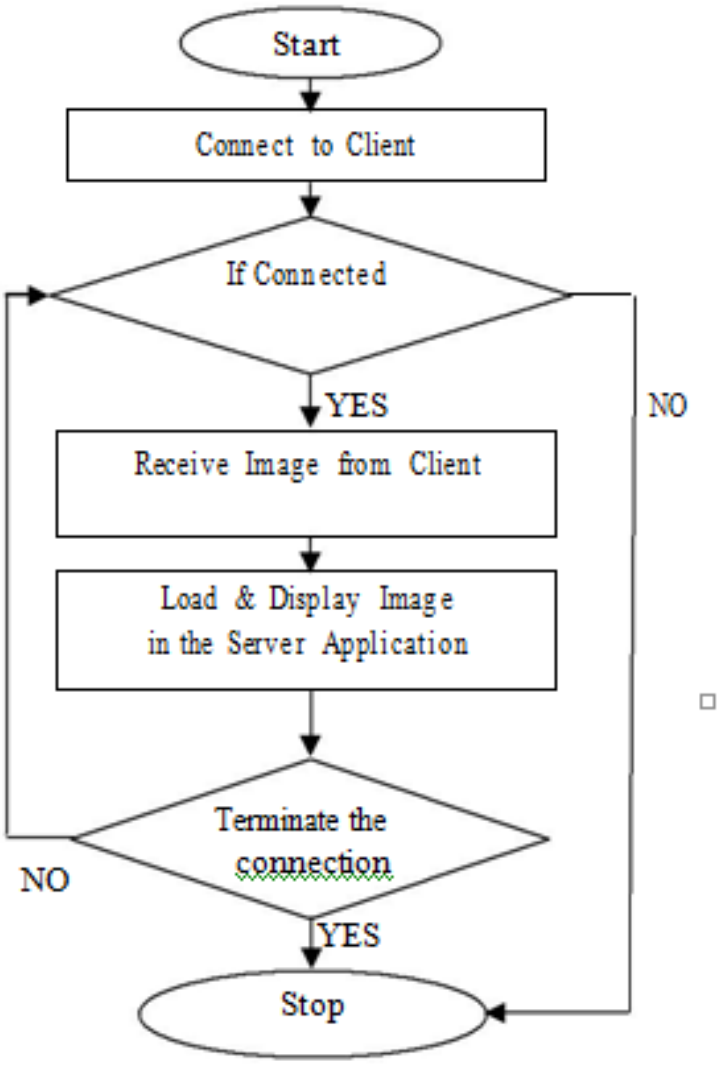
DISPLAY

FLOW LINE

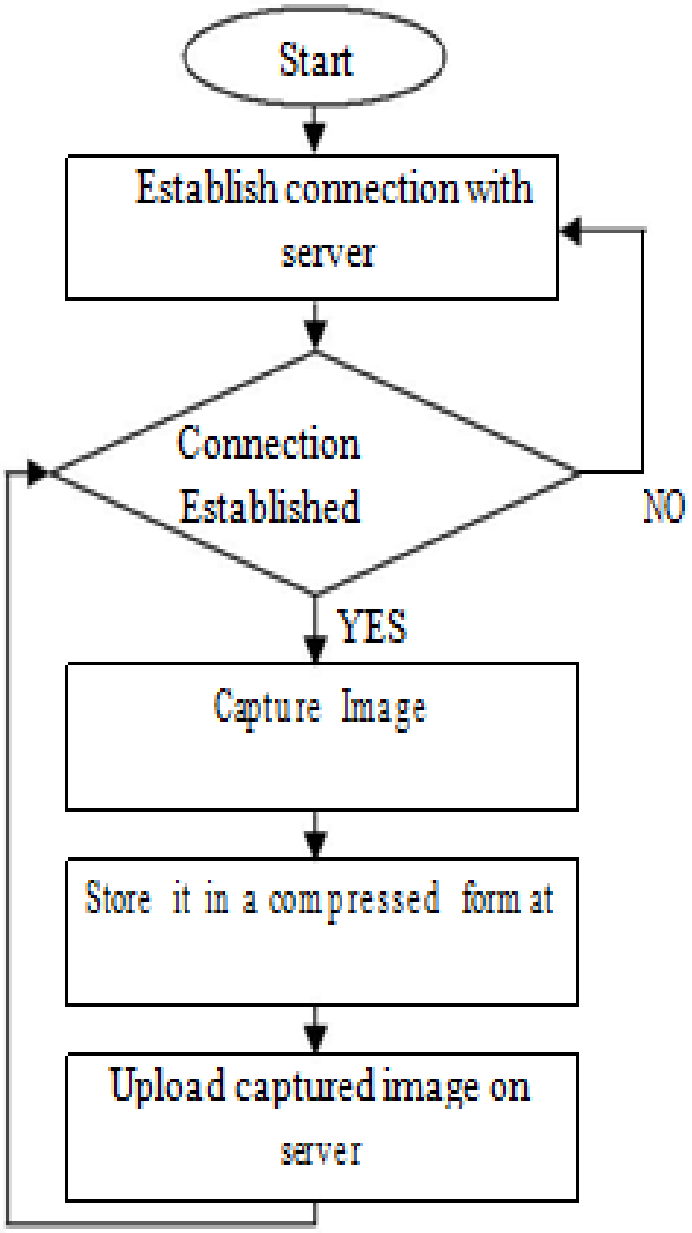
MANUAL INPUT



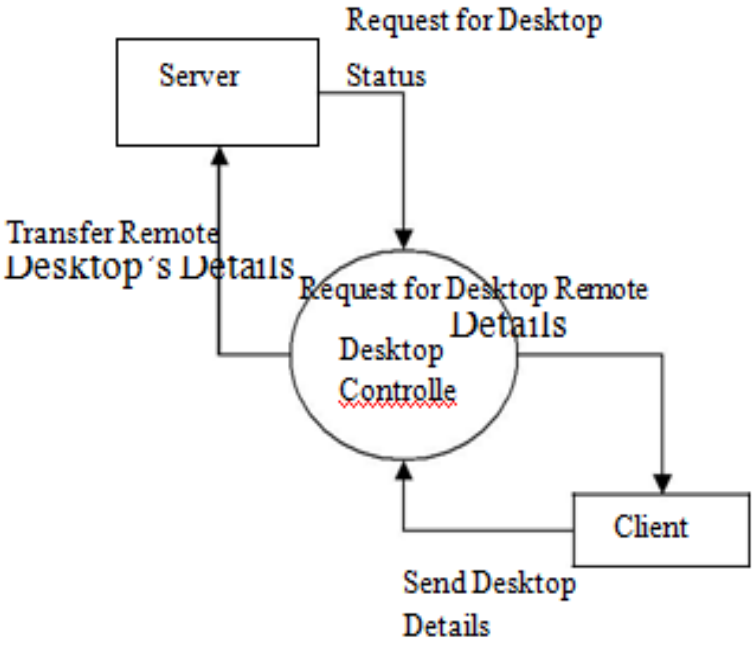
Flow Chart
Server Side



FLOW CHART Client side:



5.1 DATA FLOW DIAGRAM SERVER & CLIENT



DESIGN DETAILS

System designing is a solution, ”how to “ approach to the creation of a new system. This important phase provides the understanding and procedural details necessary for implementing the system recommend in the feasible study. The design step produces a data design, an architectural design and a procedural design.The data Design transform the information domain model created during analysis into data structures that will be required to implement the software.The architectural design defines the relationship among the major structural components into a procedural description of the software. Source code is generated and testing is conducted to integrate and validate the software.From a project Management of view, Software design is conducted in two steps.Their preliminary design is concern with the transformation in to data and Software architecture.

DATA DESIGN:-

Data design is the first of the three design activities that are conducted during software engineering. The impact of the data structure and procedural complexity causes data design to have a profound influence on the software quality. The Concepts of information hiding and data abstraction provides the foundation for an approach to data design.

ARCHITECTURAL DESIGN:-

The primary objective of the architectural design is to develop a modular program structure and represent the control relationship between modules. In addition, architectural design melds program structure and data defining interfaces that enable data to flow throughout the program.

PROCEDURAL DESIGN:

The primary design occurs after data and program structure has been established. Procedural detail can be specific sing either of the following:

- Structured programming.
- Flowchart constructs.
- Decision tables.
- Structured English.

DATA FLOWS AND THE DATA STORES

Data flows are data structures in motion, while data stores are data structures. Data flows are paths or 'pipe lines', along which data structures travel, where as the data stores are place where data structures are kept until needed. Data flows are data structures in motion, while data stores are data structures at rest. Hence it is possible that the data flow and the data store would be made up of the same data structure. Data flow diagrams is a very handy tool for the system analyst because it gives the analyst the overall picture of the system, it is a diagrammatic approach. A DFD is a pictorial representation of the path which data takes From its initial interaction with the existing system until it completes any interaction. The diagram will describe the logical data flows dealing the movements of any physical items. The DFD also gives the insight into the data that is used in the system i.e., who actually uses it is temporarily stored. A DFD does not show a sequence of steps. A DFD only shows what the different process in a system is and what data flows between them.

RULES FOR DFD:-

Fix the scope of the system by means of context diagrams.

Organize the DFD so that the main sequence of the actions reads left to right and top to bottom.

Identify all inputs and outputs.

Identify and label each process internal to the system with rounded circles.

A process is required for all the data transformation and transfers.

Therefore, never connect a data store to a data source or the destinations or another data store with just a data flow arrow.

Do not indicate hardware and ignore control information.

Make sure the names of the processes accurately convey everything the process is done.

There must not be unnamed process.

Indicate external sources and destinations of the data, with squares.

Number each occurrence of repeated external entities.

Identify all data flows for each process step, except simple Record retrievals.

Label data flow on each arrow.

Use details flow on each arrow.

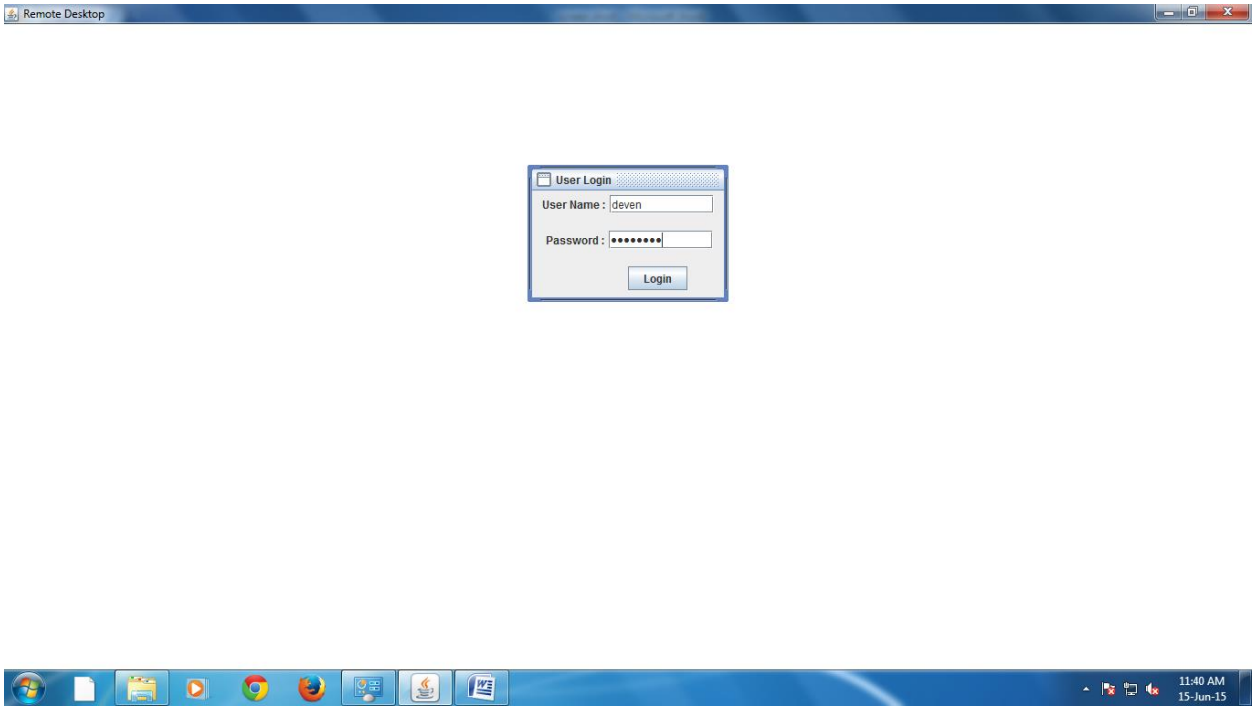
Use the details flow arrow to indicate data movements.

There can’t be unnamed data flow.

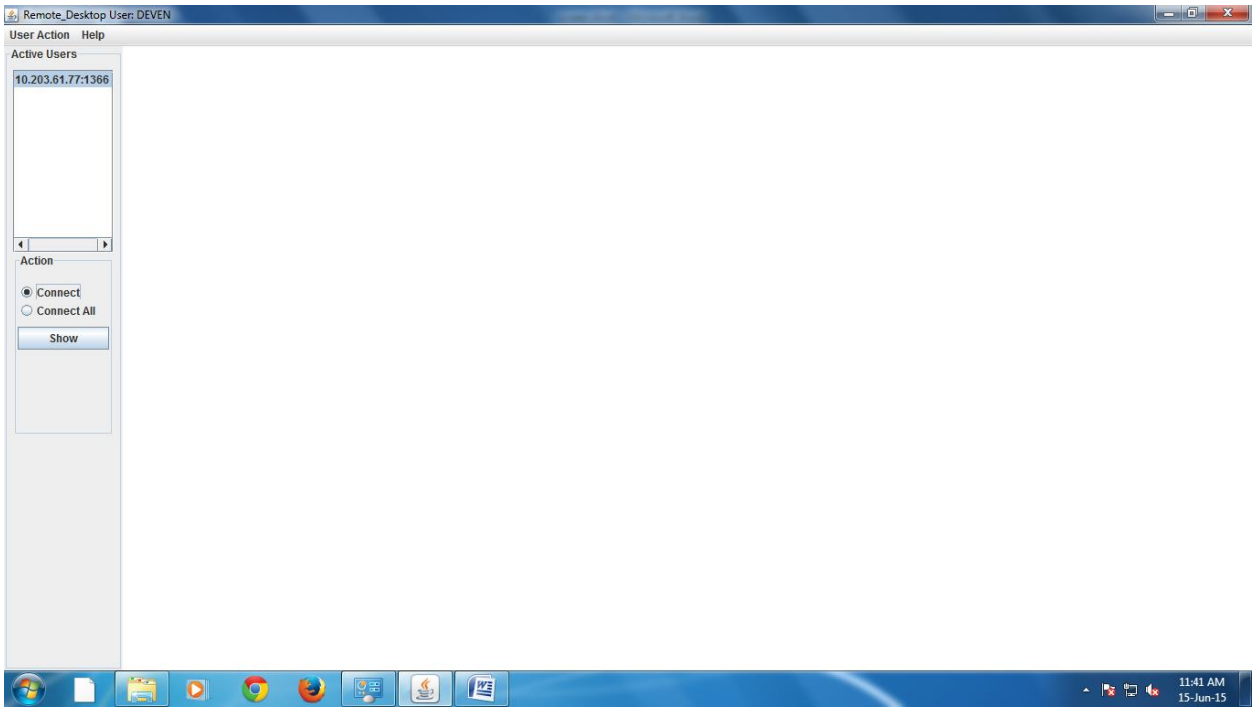
A data flow can’t connect two external entities.

6 SCREEN SHORT

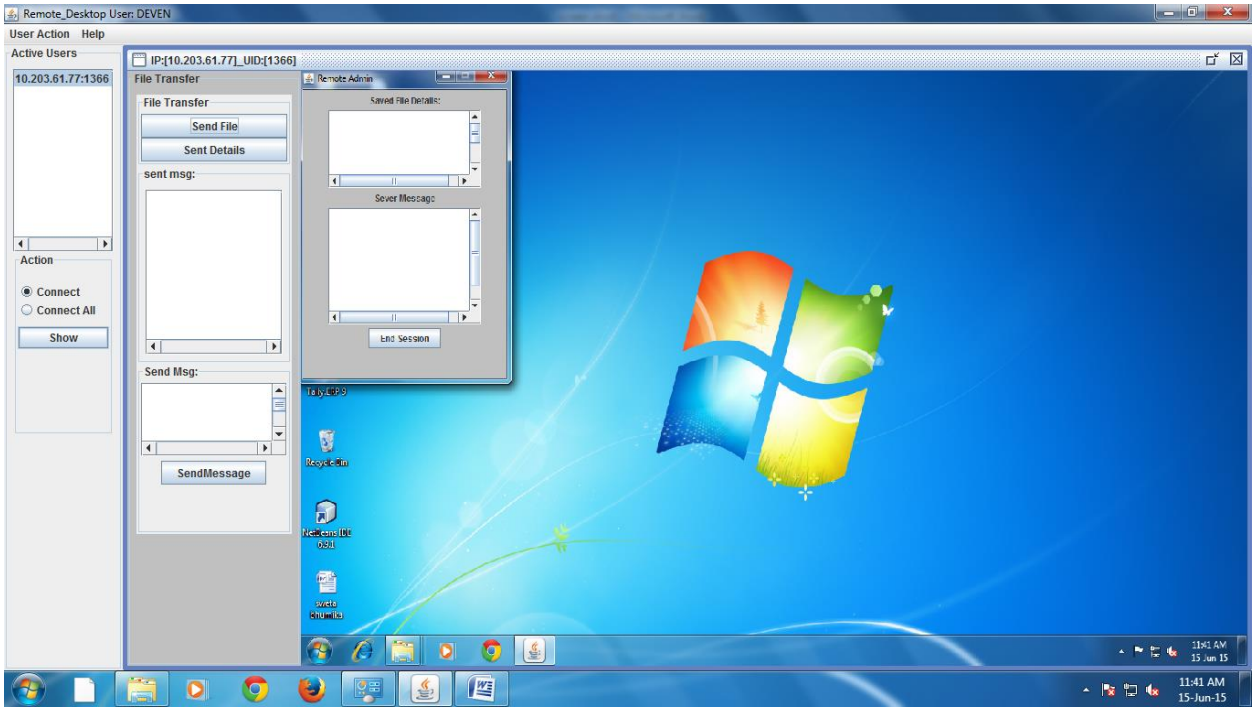
Login Screen



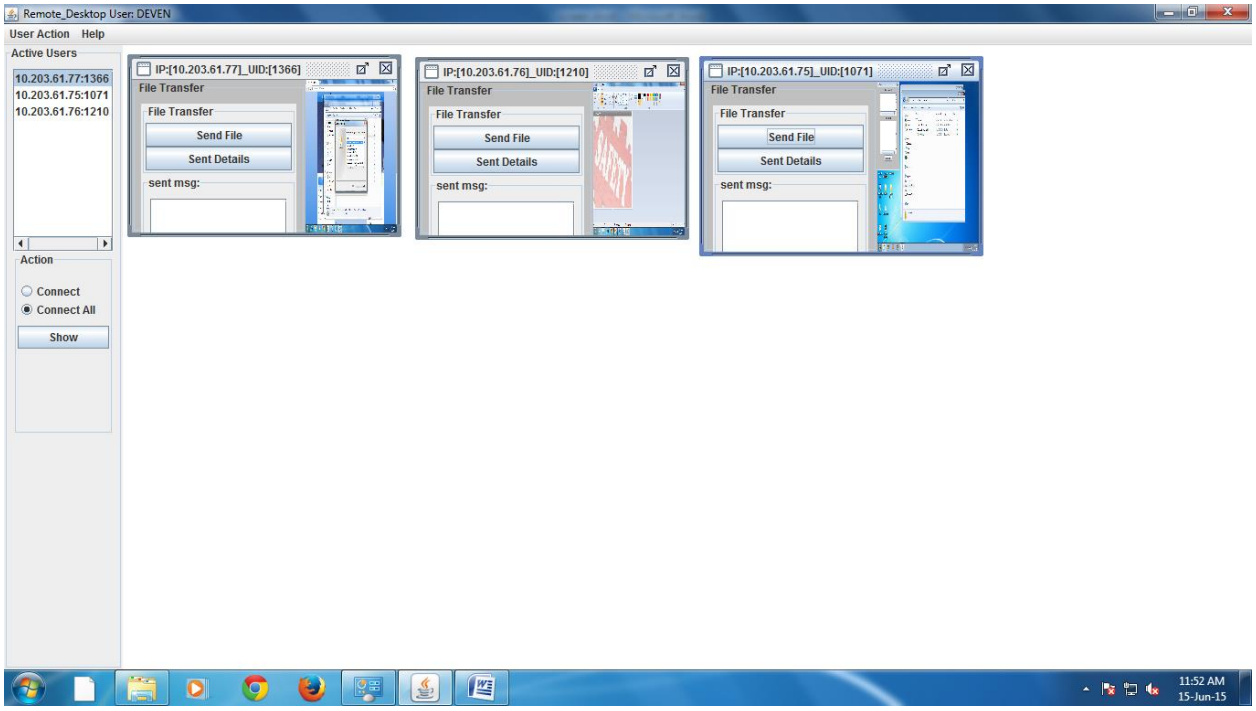
Show Active Client IP Address



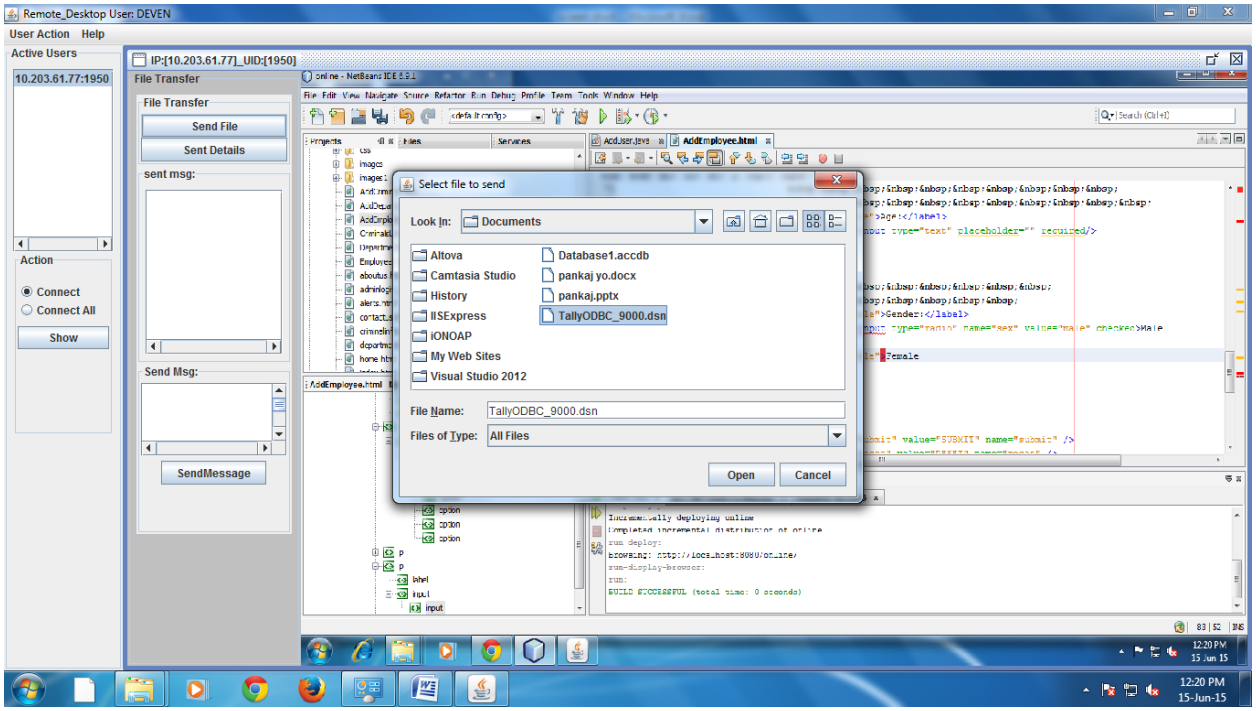
Show client Screen



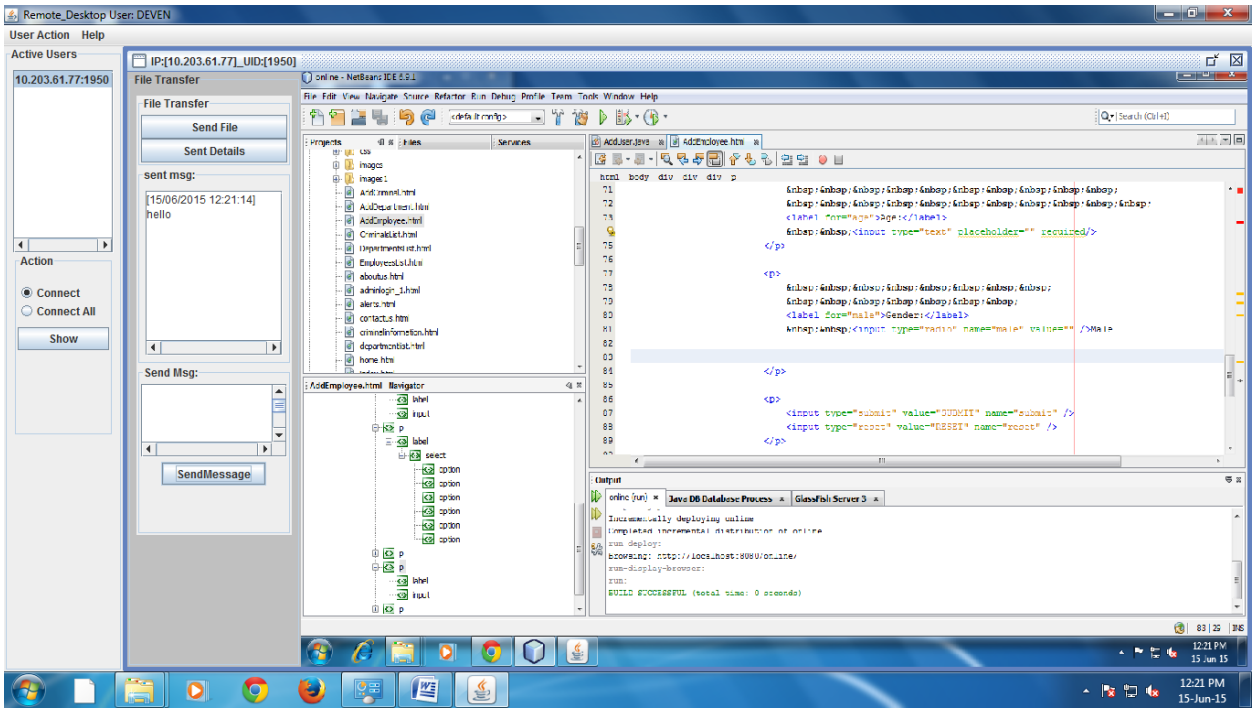
Show Multiple Client Screen



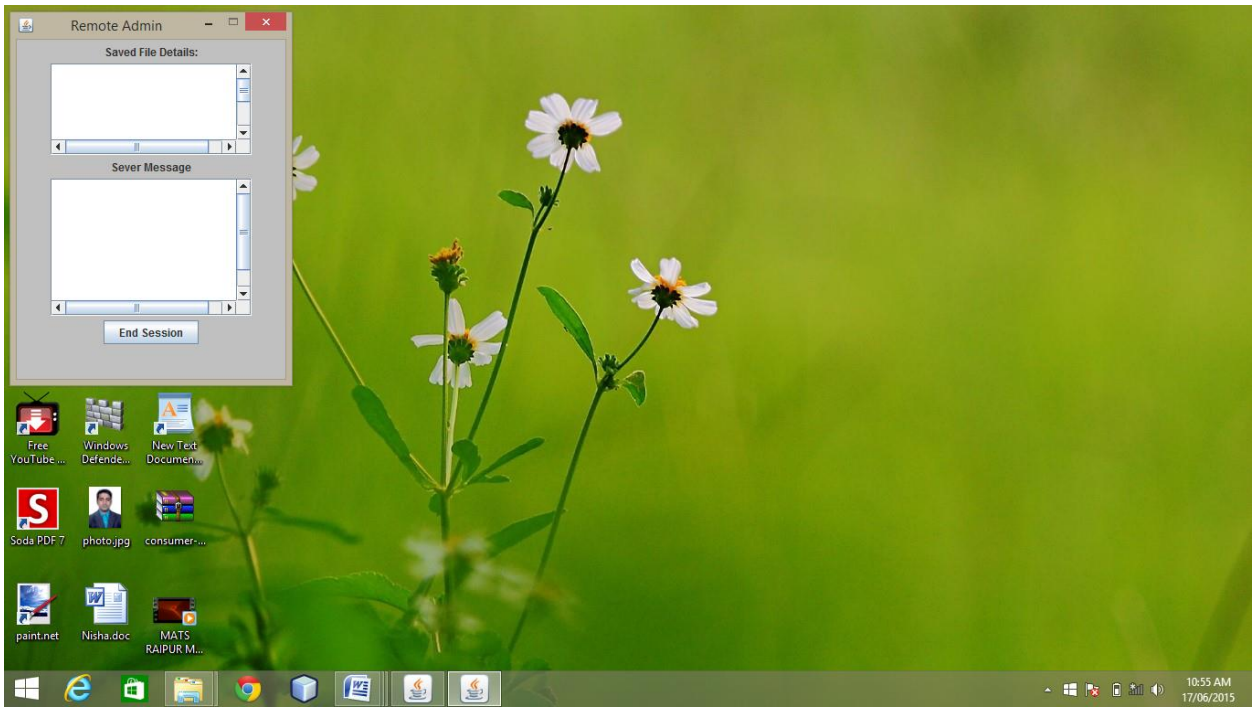
Server Send The File To Client



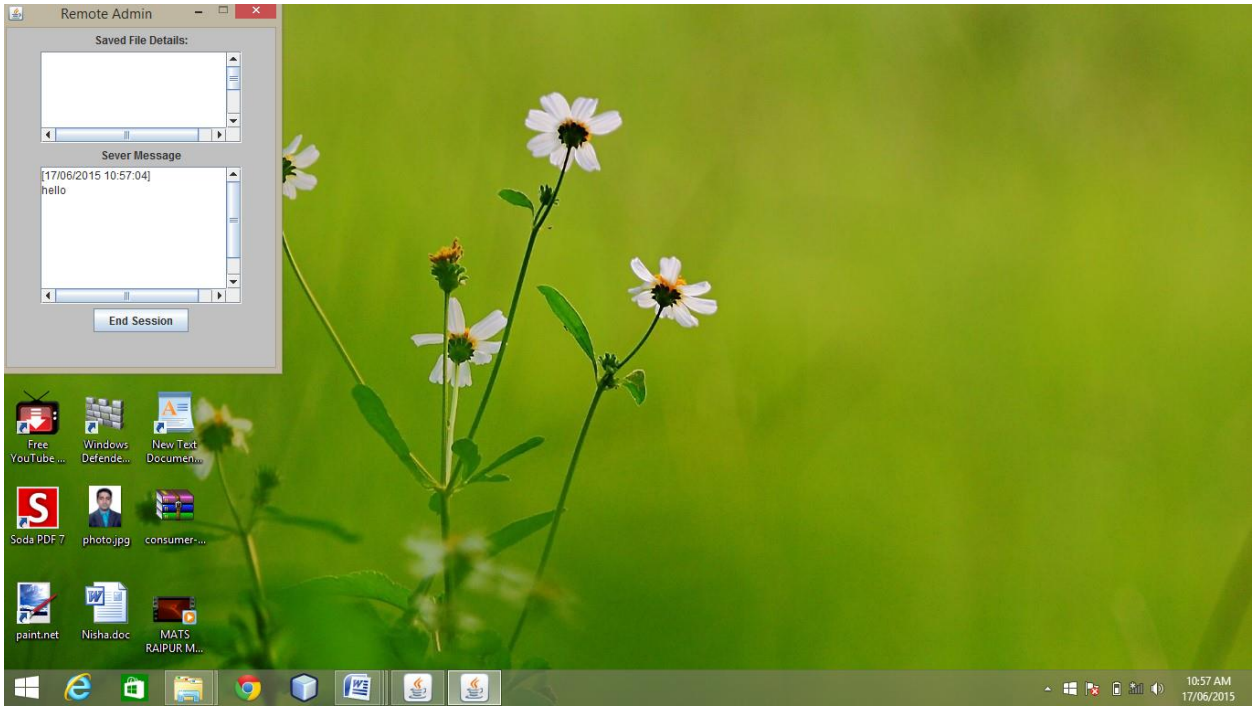
Send MSG Server to Client



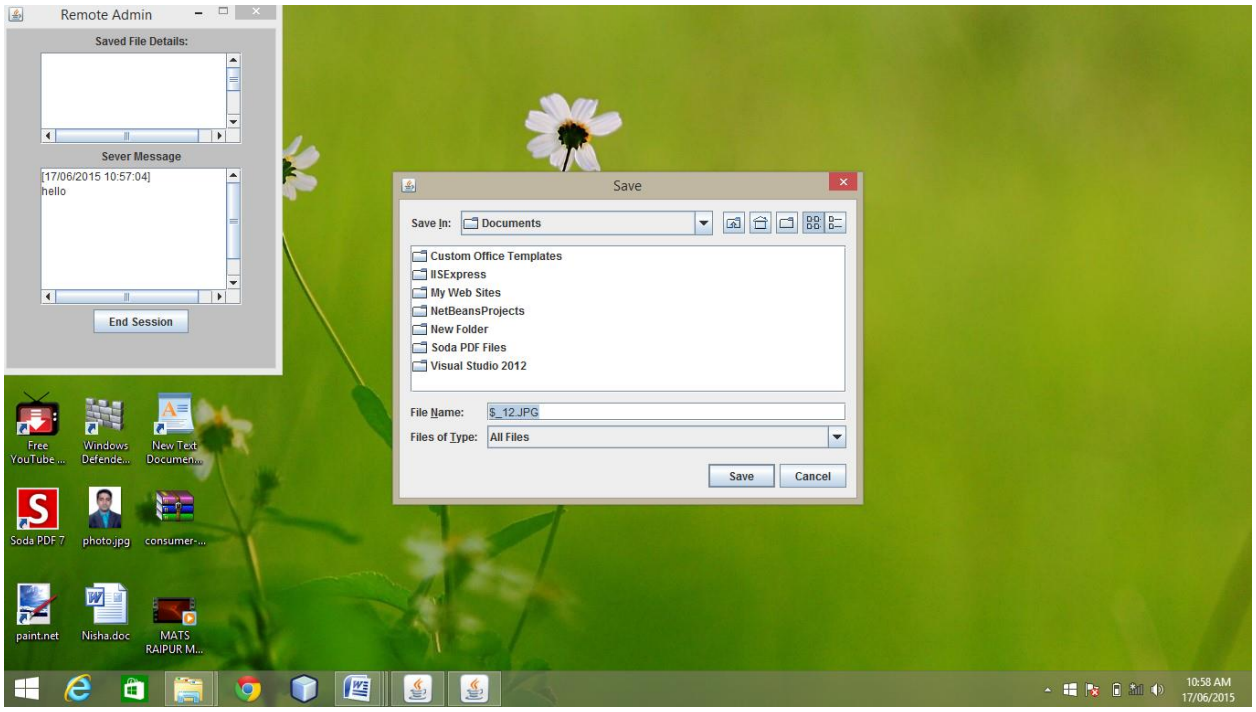
Client Screen



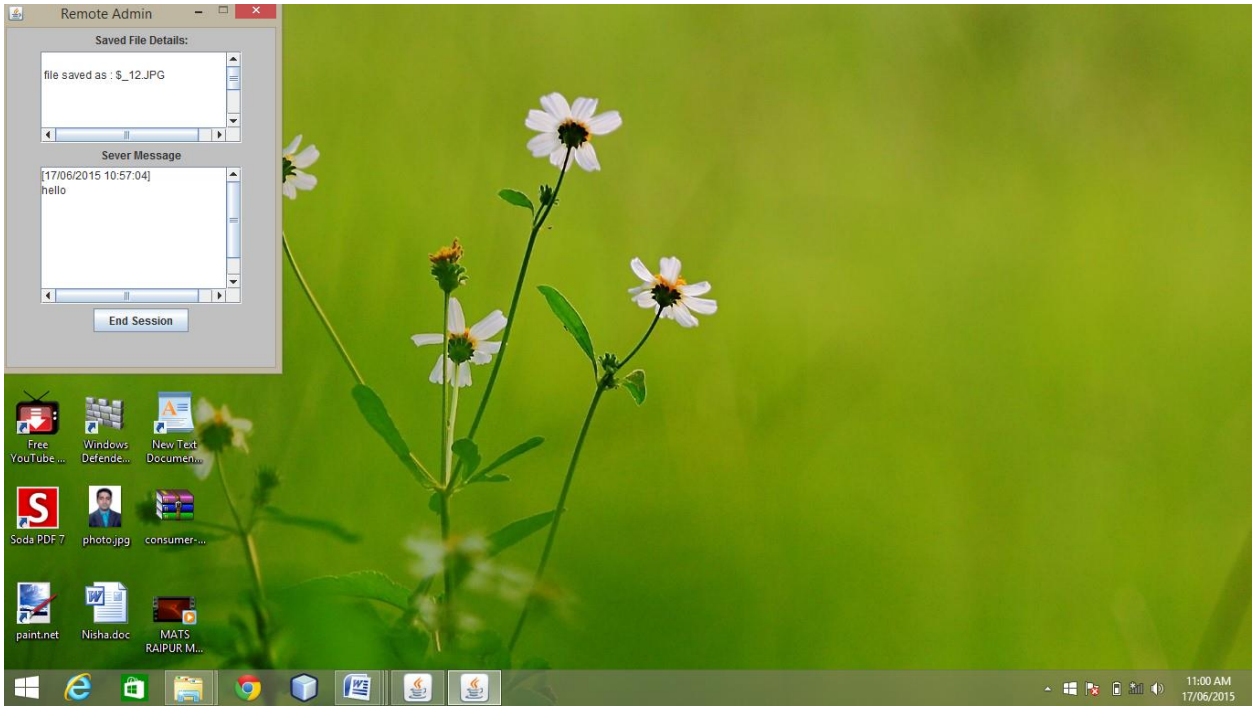
Client Receive MSG



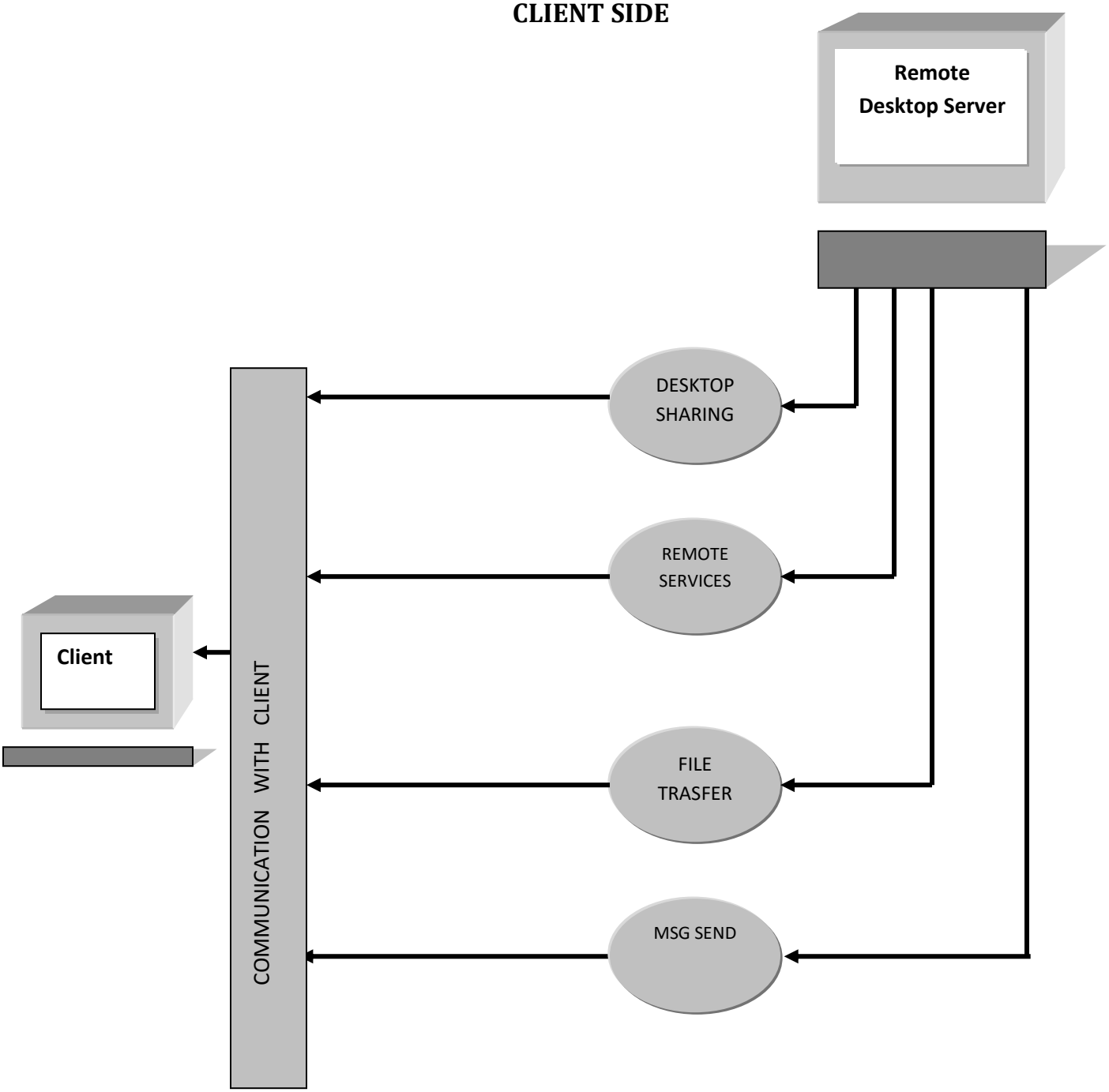
Client Receive File



Show Detail to Receive File and MSG



CLIENT SIDE


Remote
Desktop Server
DESKTOP
SHARING
REMOTE
SERVICES
FILE
TRASFER
MSG SEND
COMMUNICATION WITH CLIENT
Client

Diagrams in UML :

Diagrams are graphical presentation of set of elements. Diagrams visualize a system from different angles and perspectives.

The UML has Nine diagrams these diagrams can be classified into the following groups.

Static:

1. Class diagrams.
2. Object diagrams.
3. Component diagrams.
4. Deployment diagrams

Dynamic:

1. Use case diagram.
2. Sequence diagram.
3. Collaboration diagram.
4. State chart diagram.
5. Activity diagram.

Static or structural diagrams :

Class diagram :

This shows a set of classes, interfaces, collaborations and their relationships. There are the most common diagrams in modeling the object oriented systems and are used to give the static view of a system.

Object diagram:

Shows a set of objects and their relationships and are used to show the data structures, the static snapshots of instances of the elements in a class diagram. Like class diagram, the object diagrams also address the static design view or process view of a system.

Component diagram:

Shows a set of components and their relationships and are used to illustrate the static implementation view of a system. They are related to class diagrams where in components map to one or more classes, interfaces of collaborations.

Deployment diagram:

Shows a set of nodes and their relationships. They are used to show the static deployment view of the architecture of a system.

They are related to the component diagrams where a node encloses one or more components.

Dynamic or behavioral diagrams:

Use Case diagram:

Shows a set of use cases and actors and their relationships. These diagrams illustrate the static use case view of a system and are important in organizing and modeling the behaviors of a system.

Sequence diagram & collaboration diagram:

These two diagrams are semantically same i.e. the dynamics of a system can be modeled using one diagram and transform it to the other kind of diagram without loss of information. Both form the, Interaction diagram.

Sequence diagram:

Sequence diagram is an interaction diagram which focuses on the time ordering of messages it shows a set of objects and messages exchange between these objects. This diagram illustrates the dynamic view of a system.

collaboration diagram:

This diagram is an interaction diagram that stresses or emphasizes the structural organization of the objects that send and receive messages. It shows a set of objects, links between objects and messages send and received by those objects. There are used to illustrate the dynamic vies of a system.

State Chart Diagram and Activity Diagrams:

These Diagrams are semantically similar. State chart diagram shows a state machine consisting of states, transitions and activities these illustrates the dynamic view of a system. They focuses on the event ordered Behavior of an object.

Activity Diagrams:

Activity diagram shows the flow from one activity to another within a system. The activities may be sequential or branching objects that act and are acted upon. These also show the dynamic view of the system.

7. CODING

Remote Server

ServerInitiator Class

This is the entry class which listens to server port and wait for clients connections. Also, it creates an essential part of the program GUI.

ClientHandler Class

Per each connected client, there is an object of this class. It shows an InternalFrame per client and it receives clients' screen dimension.

ClientScreenReciever Class

Receives captured screen from the client, then displays it.

ClientCommandsSender Class

It listens to the server commands, then sends them to the client. Server commands include mouse move, key stroke, mouse click, etc.

EnumCommands Class

Defines constants which are used to represent server commands.

Remote Server_source:-

```
package remoteserver;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyVetoException;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

import javax.swing.BorderFactory;
import javax.swing.ButtonGroup;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JDesktopPane;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
```

```
import remoteserver.user.User;
import remoteserver.utility.ConnectionUtility;
/**
 * @author  Devendra kumar
 * @author Surendra Kumar
 * @version 1.0
 * @since june 2015
 *  Raipur:C.G.*/



public class ServerInitiator {

        ServerSocket sc;
        ServerSocket filesc;
   //Main server frame
   private JFrame frame = new JFrame("Remote Desktop");
   //JDesktopPane represents the main container that will contain all
   //connected clients' screens
   private JDesktopPane desktop = new JDesktopPane();

    static int totalLoginAttempt=0;

    static User systemUser;
    JPanel activeUserPanle =  new JPanel();

    JMenuBar menuBar = new JMenuBar();
    JMenu userActionsMenu  =new JMenu("User Actions");
    JMenuItem resetPassword  = new JMenuItem("Reset Password");
    JMenuItem addUser  = new JMenuItem("Add User");
    JMenu helpMenu = new JMenu("Help");
    JMenuItem about  = new JMenuItem("About Us");



        static Map<String,JInternalFrame> clientsMap = new HashMap<String,
JInternalFrame>();
        static DefaultListModel listModel = new DefaultListModel();
        static JList activeClientJList = new JList(listModel);

        static JRadioButton singConnectRadio = new JRadioButton("Connect");
        static JRadioButton allConnectRadio = new JRadioButton("Connect All");
        ButtonGroup connectRadioGroupBtn  = new ButtonGroup();
        JButton showActiveUserBtn = new JButton("Show");
```

```
public static void main(String args[]){
   int serverPort = ServerInitiator.getServerPort();
   new ServerInitiator().initialize(serverPort);

}


      public void initialize( int port){

      //initialize Server start
      try {
                        sc = new ServerSocket(port);
            } catch (Exception e) {
                        JOptionPane.showMessageDialog(null,"ServerError:"+ .getMessage());
                        System.exit(0);
            }
            try {
                        //file server port will be port+1;
                        filesc = new ServerSocket(port+1);
            } catch (Exception e) {
                        JOptionPane.showMessageDialog(null,"ServerError:"+ .getMessage());
                        System.exit(0);
            }finally{
            }
            //initialize Server End

            //if success fully server started then go for next step


       try {
      //----------------------------

      //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");
         //as soon as remote client get connect it will
         //againg waiting for file click
         Socket fileClientSocket = filesc.accept();
         System.out.println("New FILE client Connected to the server");
```

```
            //Per each client create a ClientHandler
            new ClientHandler(client,fileClientSocket,desktop);
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

/*
 * Draws the main server GUI
 */
public void drawGUI(){


        activeUserPanle.setBorder(BorderFactory.createTitledBorder("Active Users"));
        //activeUserPanle.setBackground(Color.red);
        activeUserPanle.setSize(200, activeUserPanle.getHeight());
        activeUserPanle.setVisible(false);

        JPanel activeClientListPanel = new JPanel(new GridLayout(2,1));
        activeClientListPanel.setBackground(Color.GREEN);
                JScrollPane activeClientListScroll = new JScrollPane(activeClientJList,
                                        JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
                                        JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
                activeClientListScroll.setPreferredSize(new Dimension(110, 200));


                JPanel userSelectionPanel = new JPanel(new GridLayout(2,1));
                userSelectionPanel.setBorder(BorderFactory.createTitledBorder("Action"));
                JPanel userSelectionTopPanel = new JPanel();
                userSelectionTopPanel.setLayout(null);
                //userSelectionTopPanel.setBackground(Color.CYAN);

                connectRadioGroupBtn.add(singConnectRadio);
                connectRadioGroupBtn.add(allConnectRadio);
                singConnectRadio.setBounds(0, 10, 110, 25);
                allConnectRadio.setBounds(0, 30, 110, 25);
                showActiveUserBtn.setBounds(0, 60, 100, 25);
                userSelectionTopPanel.add(singConnectRadio);
                userSelectionTopPanel.add(allConnectRadio);
                userSelectionTopPanel.add(showActiveUserBtn);
                userSelectionPanel.add(userSelectionTopPanel);

        activeClientListPanel.add(activeClientListScroll);
        activeClientListPanel.add(userSelectionPanel);
```

```
    activeUserPanle.add(activeClientListPanel);

            frame.add(activeUserPanle,BorderLayout.WEST);
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);
frame.addWindowListener(new WindowAdapter(){
    @Override
    public void windowClosing(WindowEvent e) {

            try {
                    ConnectionUtility.closeConnection();
                            } catch (Exception exception) {
                            }

            System.exit(0);
    }
});


showActiveUserBtn.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {

            //perform only if any active client user is present


            if(listModel.size()<=0){
JOptionPane.showMessageDialog(frame, "No active client To perform action.");
                    return;
            }

            //perform only if any selection is present
            if(!(singConnectRadio.isSelected() || allConnectRadio.isSelected())){
                    JOptionPane.showMessageDialog(frame, "plz select any action type ");
                    return;
            }


            //process for all active client
            if(allConnectRadio.isSelected()){
                    int cascade=10;
                    int x=5;
```

```
                for(JInternalFrame tempJIFrame:clientsMap.values()){
                try {
                        tempJIFrame.setVisible(true);
                        tempJIFrame.show();
                        tempJIFrame.setFocusable(true);

                                tempJIFrame.setMaximum(false);
                                tempJIFrame.setSize(300, 200);
                                tempJIFrame.setLocation(x, cascade);
                                x = x+ 10;
                                cascade=cascade+30;


                        } catch (PropertyVetoException ex) {
                        }
        }

                return ;

        }//end of all selection


        //to handle sigle client
        if(singConnectRadio.isSelected()){
                //if no list item selected then give alert
                if(activeClientJList.getSelectedValue()== null){

JOptionPane.showMessageDialog(frame, "Please select active client first.");
                        return ;
                }

                //hide all first
                for(JInternalFrame tempJIFrame:clientsMap.values()){
                        try {
                                tempJIFrame.setVisible(false);
                                tempJIFrame.hide();
                                tempJIFrame.setFocusable(false);
                                tempJIFrame.setMaximum(false);
                        } catch (PropertyVetoException ex) {
                        }
                }


JinternalFrameinterFrame=clientsMap.get(activeClientJList.getSelectedValue());
```

```
                    if(interFrame==null){
                     //no such active user
                            return;
                    }

                    try {
                    interFrame.setVisible(true);
                    interFrame.setFocusable(true);
                                    interFrame.setMaximum(true);
                            } catch (PropertyVetoException ex) {
                                    ex.printStackTrace();
                            }


                }// end if  of singConnectRadio.isSelected()




            }//end of actionPerformed
        });//end of addActionListener

        /*
         * after create GUI it will process for login
         */
           processLogin();

    }

    private void processLogin() {
            final JInternalFrame interFrame = new JInternalFrame("", true, true, true);
// TODO Auto-generated method stub

             interFrame.setLayout(new BorderLayout());
               JPanel userPanel = new JPanel();
                   interFrame.getContentPane().add(userPanel ,BorderLayout.CENTER);

               //setting title of interFrame

               String title = "User Login";
                            interFrame.setTitle(title);
                            interFrame.setSize(220, 150);

                       Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
                       int x = (int) ((dimension.getWidth() - interFrame.getWidth()) / 2);
```

```
        int y = (int) ((dimension.getHeight() - interFrame.getHeight()) / 4);

        interFrame.setLocation(x, y);
            //interFrame.setBounds(300, 100, 220, 150);
            //add interFrame to desktop
    desktop.add(interFrame);

    //this allows to handle KeyListener events
    userPanel.setFocusable(true);
    //userPanel.setBorder(BorderFactory.createTitledBorder("Client Sceen"));
    interFrame.setVisible(true);
    // interFrame.setFocusable(true);
    interFrame.setMaximizable(false);
    interFrame.setClosable(false);

    JPanel namePanel = new JPanel();
    JPanel passPanel = new JPanel();
    JPanel submitPanel = new JPanel();

    final JTextField userTxt = new JTextField(10);
    final JPasswordField passTxt = new JPasswordField(10);
    JButton loginBtn = new JButton("Login");

    namePanel.add(new JLabel("User Name :"));
    namePanel.add(userTxt);
    userTxt.setFocusable(true);
    passPanel.add(new JLabel(" Password :"));
    passPanel.add(passTxt);

    submitPanel.add(new JLabel("                "));
            submitPanel.add(loginBtn );

    userPanel.setLayout(new GridLayout(3,1));
    userPanel.add(namePanel);
    userPanel.add(passPanel);
    userPanel.add(submitPanel);
    //userPanel.setBackground(Color.CYAN);
    interFrame.setSize(interFrame.getSize());

    loginBtn.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {

            //------------------------
            if(totalLoginAttempt>=3){
JOptionPane.showMessageDialog(interFrame, "Total number of Login Attempt Acceded for
```

```
3 Times\nPlease Re-start");
                                System.exit(0);
                        }

                        String error = "";
                                String user = userTxt.getText().trim();
                                String password = new String(passTxt.getPassword());

                                if(user.length()<=0){
                                        error ="Required :User Name";
                                }
                                if(password.length()<=0){
                                        error = error+"\nRequired :User Password";
                                }
                                if(! "".equals(error)){

        JOptionPane.showMessageDialog(interFrame,error);
                                   return;
                                }

                                // ms access databasename :RemoteAdminUser.accdb
                                //Table Name: admin_user  :
                                // field Names:  name,password
                                // Dsn NameL: RDA_DSN
                                //valide from data

                                StringBuffer dbError=new StringBuffer();
                                Connection con = null;
                                Statement stmt = null;
                                ResultSet rs = null;
                                try {
        con = ConnectionUtility.getConnection();
        stmt = con.createStatement();
        rs = stmt.executeQuery("SELECT * FROM admin_user " +"WHERE name='"+user+"'
AND password='"+password+"'");
        totalLoginAttempt++;
        //if user found it create and assign and return
                                        if(rs.next()){
systemUser = new User(user,password);
frame.setTitle("Remote_Desktop User: "+userTxt.getText().toUpperCase());
                                interFrame.dispose();
                                activeUserPanle.setVisible(true);
                                createMenuBar();

                                                return;
```

```
                                                    }else{
dbError.append("Error:No such User Or password found in Database");
                                                    }

                    } catch (SQLException ee) {
                    dbError.append("\nError:"+ee.getMessage());
                    }catch (Exception eee) {
                    dbError.append("\nError:"+eee.getMessage());
                    }
                    if(! "".equals(dbError)){

          JOptionPane.showMessageDialog(interFrame,dbError);
                                              }

                    }

                });


          }



private void createMenuBar() {
                    // TODO Auto-generated method stub
                    menuBar = new JMenuBar();
                    menuBar.setSize(200, 400);
                    userActionsMenu = new JMenu("User Action");
                    addUser = new JMenuItem("Add User");
               resetPassword  = new JMenuItem("Reset Password");
               helpMenu = new JMenu("Help");
               helpMenu.add(about);

               userActionsMenu.add(addUser);
                    userActionsMenu.add(resetPassword);

                    menuBar.add(userActionsMenu);
                    menuBar.add(helpMenu);
                    frame.setJMenuBar(menuBar);

                    addUser.addActionListener(new ActionListener(){
                              public void actionPerformed(ActionEvent e) {

  final JInternalFrame interFrame = new JInternalFrame("", true, true, true);
// TODO Auto-generated method stub
```

```
                        interFrame.setLayout(new BorderLayout());
                          JPanel userPanel = new JPanel();
interFrame.getContentPane().add(userPanel ,BorderLayout.CENTER);

                          //setting title of interFrame

                          String title = "Add User";
                                    interFrame.setTitle(title);
                                    interFrame.setSize(220, 150);

  Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
  int x = (int) ((dimension.getWidth() - interFrame.getWidth()) / 2);
  int y = (int) ((dimension.getHeight() - interFrame.getHeight()) / 4);

                               interFrame.setLocation(x, y);
                                    //interFrame.setBounds(300, 100, 220, 150);
                                    //add interFrame to desktop
                          desktop.add(interFrame);

                          //this allows to handle KeyListener events
                          userPanel.setFocusable(true);
                          //userPanel.setBorder(BorderFactory.createTitledBorder("Client
Sceen"));
                          interFrame.setVisible(true);
                         // interFrame.setFocusable(true);
                          interFrame.setMaximizable(false);
                          interFrame.setClosable(true);

                          JPanel namePanel = new JPanel();
                          JPanel passPanel = new JPanel();
                          JPanel submitPanel = new JPanel();

                          final JTextField userTxt = new JTextField(10);
                          final JPasswordField passTxt = new JPasswordField(10);
                          JButton AddUserBtn = new JButton("Add User");

                          namePanel.add(new JLabel("User Name :"));
                          namePanel.add(userTxt);
                          userTxt.setFocusable(true);
                          passPanel.add(new JLabel(" Password :"));
                          passPanel.add(passTxt);

                          submitPanel.add(new JLabel("                 "));
                                    submitPanel.add(AddUserBtn );
```

```
                userPanel.setLayout(new GridLayout(3,1));
                userPanel.add(namePanel);
                userPanel.add(passPanel);
                userPanel.add(submitPanel);
                //userPanel.setBackground(Color.CYAN);
                interFrame.setSize(interFrame.getSize());

                AddUserBtn.addActionListener(new ActionListener(){
                    public void actionPerformed(ActionEvent e) {

                String error = "";
                                        String user = userTxt.getText().trim();
                    String password = new String(passTxt.getPassword());

                                        if(user.length()<=0){
                                                error ="Required :User Name";
                                        }
                                        if(password.length()<=0){
                    error = error+"\nRequired :User Password";
                                        }
                                        if(! "".equals(error)){

        JOptionPane.showMessageDialog(interFrame,error);
                                           return;
                                        }

                        // ms access databasename :RemoteAdminUser.accdb
                                        //Table Name: admin_user  :
                                        // field Names:  name,password
                                        // Dsn NameL: RDA_DSN
                                        //valide from data

                                        StringBuffer dbError=new StringBuffer();
                                        Connection con = null;
                                        Statement stmt = null;
                                        try {
                                    con = ConnectionUtility.getConnection();
                                                stmt = con.createStatement();


ResultSet rs = stmt.executeQuery("SELECT*FROM admin_user " +"WHERE name='"+user+"'");
        //if user found it create and assign and return
                                        if(rs.next()){
```

```
        JOptionPane.showMessageDialog(interFrame,   " user :" +user+" is already exist");
                                                                    return;
                                                                }

int updated = stmt.executeUpdate("insert into admin_user (name,password) "+"values('"+user+"','"+password+"')");
//if user found it create and assign and return
                                                                if(updated>=1){

        JOptionPane.showMessageDialog(interFrame, "user :" +user+" added successfuly");
                                                                    interFrame.dispose();
                                                                    return;
                                                                }

                                                            } catch (SQLException ee) {

        dbError.append("\nError:"+ee.getMessage());
                                                            }catch (Exception eee) {

        dbError.append("\nError:"+eee.getMessage());
                                                            }
                                                            if(! "".equals(dbError)){

        JOptionPane.showMessageDialog(interFrame,dbError);
                                                            }

                                    }//end actionPerformed
                                });//end listern


                            }
                    });//End of addUser.addActionListener

                    resetPassword.addActionListener(new ActionListener(){

                            public void actionPerformed(ActionEvent e) {

                     final JInternalFrame interFrame = new JInternalFrame("", true, true, true);

                                interFrame.setLayout(new BorderLayout());
                                  JPanel userPanel = new JPanel();
                                interFrame.getContentPane().add(userPanel ,BorderLayout.CENTER);

                                    //setting title of interFrame
```

```
                String title = "Reset User";
                        interFrame.setTitle(title);
                        interFrame.setSize(220, 150);

            Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
           int x = (int) ((dimension.getWidth() - interFrame.getWidth()) / 2);
int y = (int) ((dimension.getHeight() - interFrame.getHeight()) / 4);

                    interFrame.setLocation(x, y);
                        //interFrame.setBounds(300, 100, 220, 150);
                        //add interFrame to desktop
                desktop.add(interFrame);

                //this allows to handle KeyListener events
                userPanel.setFocusable(true);
 //userPanel.setBorder(BorderFactory.createTitledBorder("Client Sceen"));
                interFrame.setVisible(true);
                // interFrame.setFocusable(true);
                interFrame.setMaximizable(false);
                interFrame.setClosable(true);

                JPanel namePanel = new JPanel();
                JPanel passPanel = new JPanel();
                JPanel submitPanel = new JPanel();

                final JTextField userTxt = new JTextField(10);
                final JPasswordField passTxt = new JPasswordField(10);
                JButton resetPasswordBtn = new JButton("Reset Password");

                namePanel.add(new JLabel("User Name :"));
                namePanel.add(userTxt);
                userTxt.setText(systemUser.getUserName());
                userTxt.setEditable(false);

                passPanel.add(new JLabel(" Password :"));
                passPanel.add(passTxt);
                passTxt.setText(systemUser.getUserPassword());
                submitPanel.add(new JLabel("                "));
                        submitPanel.add(resetPasswordBtn );

                userPanel.setLayout(new GridLayout(3,1));
                userPanel.add(namePanel);
                userPanel.add(passPanel);
                userPanel.add(submitPanel);
```

```
                        //userPanel.setBackground(Color.CYAN);
                        interFrame.setSize(interFrame.getSize());

                        resetPasswordBtn.addActionListener(new ActionListener(){
                            public void actionPerformed(ActionEvent e) {

                                String error = "";
                                        String user = userTxt.getText().trim();
                                String password = new String(passTxt.getPassword());

                                        if(user.length()<=0){
                                                error ="Required :User Name";
                                        }
                                        if(password.length()<=0){
                                error = error+"\nRequired :User Password";
                                        }
                                        if(! "".equals(error)){

        JOptionPane.showMessageDialog(interFrame,error);
                                                return;
                                        }

                                // ms access databasename :RemoteAdminUser.accdb
                                //Table Name: admin_user  :
                                // field Names:  name,password
                                // Dsn NameL: RDA_DSN
                                //valide from data

                                        StringBuffer dbError=new StringBuffer();
                                        Connection con = null;
                                        Statement stmt = null;
                                        try {
                                        con = ConnectionUtility.getConnection();
                                                stmt = con.createStatement();
int updated = stmt.executeUpdate("update admin_user " +"set password='" +password+"'"+"WHERE name='"+user+"'");
                        //if user found it create and assign and return
                                                if(updated>=1){
JOptionPane.showMessageDialog(interFrame, "user :" +user+"\'s password reset successfuly");
                interFrame.dispose();
        return;
                }

        } catch (SQLException ee) {
```

```
dbError.append("\nError:"+ee.getMessage());
                                                }catch (Exception eee) {

dbError.append("\nError:"+eee.getMessage());
                                                }
                                                if(! "".equals(dbError)){

JOptionPane.showMessageDialog(interFrame,dbError);
                                                }

                        }//end actionPerformed
                    });//end listern
                }


        });//end of resetPassword.addActionListener

        about.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e) {
                        //==============================

Object message="" + "Remote Desktop Access\n" +"Version: 1.0\n" +"This product developed by :\n" + "Devendra Kumar Sahu\n" +"\n" +"\n" +"********SIS Raipur(C.G)*******" ;

JOptionPane.showMessageDialog(frame, message, "About Us", JOptionPane.INFORMATION_MESSAGE);

                        /*JFrame aboutFrame = new JFrame("About");
                        aboutFrame.setAlwaysOnTop(true);
                        aboutFrame.setVisible(true);
                        aboutFrame.setResizable(false);
                        aboutFrame.setVisible(true);

aboutFrame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
                        aboutFrame.setSize(300, 200); */
                        //==============================
                }
        });

}


//read port form file //user
```

```
    public static int getServerPort() {
        int serverPort = 0;
        String SERVER_PORT = null;

        Properties serverProperties  = new Properties();
                Reader inStream2;
                try {
                        inStream2 = new FileReader("ServerConfig.properties");
                        serverProperties.load(inStream2);
                        SERVER_PORT = serverProperties.getProperty("SERVER_PORT");
                System.out.println("ServerConfig.properties Details\n"+serverProperties);
                } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Can not read listener port from file");
                        e.printStackTrace();
                }

        if(SERVER_PORT==null || (SERVER_PORT!=null && SERVER_PORT.isEmpty())){
SERVER_PORT = JOptionPane.showInputDialog("Please enter Server listening port");
                }

                if(SERVER_PORT!=null && SERVER_PORT.length()>0){
                        try{
                          serverPort = Integer.parseInt(SERVER_PORT) ;
                        return serverPort;
                        }catch (NumberFormatException e) {
                        JOptionPane.showMessageDialog(null, "invalid Server listening port");
                        }
                }

                if(serverPort == 0){
                 JOptionPane.showMessageDialog(null, "invalid Server listening port");
                        System.exit(0);
                }
                return serverPort;
        }

}
```

Remote Client

ClientInitiator Class

This is the entry class that starts the client instance. It establishes connection to the server and creates the client GUI.

ScreenSpyer Class

Captures screen periodically and sends them to the server.

ServerDelegate Class

Receives server commands and executes them in the client PC.

EnumCommands Class

Defines constants which are used to represent server commands.

RemoteClient_source

```
/*
 */

package remoteclient;

import java.awt.AWTException;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Properties;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

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

   Socket socket = null;
   Socket fileSocket = null;

   JPanel filePanel = new JPanel();
   JTextArea msgArea =  new JTextArea(10,20);
   JTextArea serverRecievemsgArea =  new JTextArea(10,20);
   private static Properties clientProperties;

   public static void main(String[] args){
      String ip = "127.0.0.1";//JOptionPane.showInputDialog("Please enter server IP");
      //String port = JOptionPane.showInputDialog("Please enter server port");
      //new ClientInitiator().initialize(ip, Integer.parseInt(port));
      int serverPort = 9990;

      //get input from user for host and ip id using SISServerConfig
      SISServerConfig hostPort = getSISServerConfig();

      ip = hostPort.getHostIP();
      serverPort = hostPort.getPortNumber();
```

```
		new ClientInitiator().initialize(ip, serverPort );
}

private static SISServerConfig getSISServerConfig() {

		String host = null;
		String portStr = null;
		Integer serverPort = null;


		//read host and port start
	clientProperties  = new Properties();
		Reader inStream2;

		boolean isFoundPropertyReadError=false;
		try {
			inStream2 = new FileReader("ClientConfig.properties");
			clientProperties.load(inStream2);
			portStr = clientProperties.getProperty("SERVER_PORT");
			host = clientProperties.getProperty("SERVER_HOST");
		  System.out.println("ServerConfig.properties Details\n"+clientProperties);
		} catch (Exception e) {
	//JOptionPane.showMessageDialog(null, "Can not read listener port from file");
			e.printStackTrace();
			isFoundPropertyReadError = true;
		}

		//read host and port End

		if(portStr==null && host==null){
			isFoundPropertyReadError=true;
		}
		if((portStr!=null&& portStr.length()<=0)
				&& host!=null && host.length()<=0){
			isFoundPropertyReadError=true;
		}


		//if data not read from file then
		//port and host will get from user manually
		if(isFoundPropertyReadError){
		  host = JOptionPane.showInputDialog("Please enter Server Host");
	  portStr = JOptionPane.showInputDialog("Please enter Server Port");
		}
```

```
            //JOptionPane.showMessageDialog(null, host+portStr);

            try{
              serverPort = Integer.parseInt(portStr) ;
            }catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "invalid Server port Entered");
                    e.printStackTrace();
                    System.exit(0);
            }

            if(host!=null && !host.isEmpty() && serverPort!= null){
             return new SISServerConfig(host,serverPort);
            }

            return null;
    }

    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);
   //after connect with remote
   //connect with file server
   fileSocket = new Socket(ip, port+1);
   System.out.println("Connection Established.");

   //Get default screen device
   GraphicsEnvironment gEnv=GraphicsEnvironment.getLocalGraphicsEnvironment();
   GraphicsDevice gDev=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
        new ScreenSpyer(socket,robot,rectangle);
        //ServerDelegate recieves server commands and execute them
        new ServerDelegate(socket,robot);

        //after crating file panel
        //we need file handler
        new
ServerFileRecieverHandler(fileSocket,filePanel,msgArea,serverRecievemsgArea);
        //new ServerFileRecieverHandler(fileSocket,filePanel,msgArea);


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

  private void drawGUI() {
    final JFrame frame = new JFrame("Remote Admin");
    frame.setSize(300, 400);
    frame.setResizable(false);

    // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JButton button = new JButton("End Session");
    button.addActionListener( new ActionListener() {
        public void actionPerformed(ActionEvent e) {

                try{
                        socket.close();
                        fileSocket.close();
                }catch (Exception ee) {
                                        // TODO: handle exception
                                }
                System.exit(0);
        }
    }
    );

    frame.addWindowListener(new WindowAdapter(){
```

```
        @Override
        public void windowClosing(WindowEvent e) {

                try{
                        socket.close();
                        fileSocket.close();
                }catch (Exception ee) {
        // TODO: handle exception
                                }
                System.exit(0);

        }
    });

    //file start

    /* //file realated task

    filePanel.setSize(300, 400);
    filePanel.add(new JLabel("File Details:"));
    filePanel.add(msgArea);
    filePanel.setBackground(Color.LIGHT_GRAY);

    //file End

    frame.add(button,BorderLayout.NORTH);
    frame.add(filePanel,BorderLayout.CENTER);
    frame.setVisible(true);*/

    filePanel.setSize(300, 200);

    filePanel.add(new JLabel("Saved File Details:"));
    msgArea.setEditable(false);
    //msgArea.setLineWrap(true);
    //msgArea.setWrapStyleWord(true);
JScrollPane area = new JScrollPane(msgArea,
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
    area.setPreferredSize(new Dimension(220, 100));
    filePanel.add(area);
    filePanel.setBackground(Color.LIGHT_GRAY);
    //file End
```

```
        //server message
        filePanel.add(filePanel.add(new JLabel("Sever Message")));

        //serverRecievemsgArea

        serverRecievemsgArea.setEditable(false);
JScrollPane sRecieveMsgScroll = new JScrollPane(serverRecievemsgArea,
JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_ALW
AYS);
        sRecieveMsgScroll.setPreferredSize(new Dimension(220, 150));
        filePanel.add(sRecieveMsgScroll);

        // endsession button will only see if it configure enable

StringSHOW_END_SESSION_BTN=clientProperties.getProperty("SHOW_END_SESSION_BTN
");
        if("true".equalsIgnoreCase(SHOW_END_SESSION_BTN)){
          filePanel.add(button);
        }


        frame.add(filePanel,BorderLayout.CENTER);
        frame.setVisible(true);


    }
}
```

8. TESTING TECHNOLOGIES AND SECURITY MECHANISMS

This phase determine the error in the project. If there is any error then it must be removed before delivery of the project. For determining errors various 1

UNIT TESTING:

Unit testing focuses verification effort on the smallest unit of software design – the module. Using the detail design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors detected as a result is limited by the constrained scope established for unit testing. The unit test is always white box oriented, and the step can be conducted in parallel for multiple modules.

Unit testing is normally considered an adjunct to the coding step. After source level code has been developed, reviewed, and verified for correct syntax, unit test case design begins. A review of design information provides guidance for establishing test cases that are likely to uncover errors. Each test case should be coupled with an asset of expected results.

Because a module is not a stand-alone program, driver and/or stub software must be developed for each unit test. In most applications a driver is nothing more than a main program that accepts test case data passes such data to the module(to be tested), and prints the relevant results. Stubs serve to replace modules that are subordinate (called by) the module to be tested. Stub or “dummy subprogram” users the subordinate module’s interface, may do minimal data manipulation, prints verification of entry and returns.

Drivers and stubs represent overhead. That is, both are software that must be written but that is not delivered with the final software product. If drivers and stubs are kept simple, actual overhead is relatively low. Unfortunately, many modules cannot be adequately unit tested with “simple” overhead software. In such cases, complete testing can be postponed until the integration test step.

Unit testing is simplified when a module with high cohesion is designed. When only one function is addressed by a module, the number of test cases is reduced and errors can be more easily predicted and uncovered.

SYSTEM TESTING:

Software is only one element of a larger computer based system. Ultimately, software is incorporated with other system elements (e.g. new hardware,

information), and a series of system integration and validation tests are conducted. Steps taken during software design and testing can greatly improve the probability of successful software integration in the larger system.
A classics system testing problem is “finger pointing”. This occurs when a defect is uncovered, and one system element developer blames another for the problem. Rather than including in such nonsense, the software engineer should anticipate potential interfacing problems and (1) design error handling paths that test all information coming from other elements of the system.(2) conduct a series of tests that simulate bad data or other potential errors at the software interface; (3) record the results or tests to use as “evidence” if finger pointing does occur (4) participate in the planning and design of system test to ensure that software is adequately tested.
There are many types of system tests, which are worthwhile for software-based systems, as detailed hereunder:

Recovery testing is a system test that forces the software to fail in a variety of ways that verifies that recovery is properly performed.
Security testing attempts to verify that protection mechanisms built into a system will protect it from improper penetration
Stress tests are designed to confront programs with abnormal situations.
Performance testing is designed to test the run-time performance of software within the context of an integrated system.

INTEGRATION TESTING:

A neophyte in the software world might ask a seemingly legitimate question once all modules have been unit-tested. If they all work individually, why do you doubt that they’ll work when we put them together? The problem, of course, is putting them together – interfacing. Date can be lost across an interface; one module can have an inadvertent, adverse effect on anther, sub functions, when combined, may not produce the desired major function; individually acceptable imprecision may be magnified to unacceptable levels; global data structures can present problems. Sadly, the list goes on and on.
Integration testing is a systematic technique for construction the program structure while at the same time conduction test to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design. There is often a tendency to attempt non-

incremental integration; that is, to construct the program using a big bang approach. All modules are combined in advance. The entire program is tested as a whole. And chaos usually results! A set of errors are encountered. Correction is difficult because the isolation of causes is complicated by the vast expanse of the entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

Incremental integration is the antithesis of the "big bang" approach. The program is constructed and tested is small segments, where errors are easier to isolate and correct; interfaces are more likely to be tested completely, and a systematic test approach may be applied.
Integration testing can be categorized into two types, namely top-down integration or bottom-up integration. Top-down integration is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module. Modules subordinate to the main control module are incorporated into the structure in either a depth-first or breadth-first manner. The bottom-up integration testing as its name implies, begins construction and testing with atomic modules. Because modules are integrated for the bottom up processing required for modules subordinate to given level is always available and the need for stubs is eliminated.

1) The selection of an integration strategy depends upon software characteristic and, sometime project schedule. In general, a combined approach that uses the top-down strategy for the upper levels of the program structure, coupled with a bottom-up strategy for the subordinate levels, may be the best compromise.

Security Testing:

Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended. It also aims at verifying 6 basic principles as listed below:

- Confidentiality
- Integrity
- Authentication

- Authorization
- Availability
- Non-repudiation

  - It is a type of non-functional testing.
  - Security testing is basically a type of software_testing that's done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
  - It is a process to determine that an information system protects data and maintains functionality as intended.
  - The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software's and hardware's and firewall etc.
  - Software security is about making software behave in the presence of a malicious attack.
  - The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, availability, authorization and non-repudiation.

Black Box Testing:

Also known as *functional testing*. A software testing technique whereby the internal workings of the item being tested are not known by the tester. For example, in a black box test on a software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.

The advantages of this type of testing include:

- The test is unbiased because the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.

- Test cases can be designed as soon as the specifications are complete.
  The disadvantages of this type of testing include:
- The test can be redundant if the software designer has already run a test case.
- The test cases are difficult to design.
- Testing every possible input stream is unrealistic because it would take a inordinate amount of time; therefore, many program paths will go untested.

White Bok Testing:

Also known as *glass box*, *structural*, *clear box* and *open box testing*. A software testing technique where by explicit knowledge of the internal workings of the item being tested are used to select the test data. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can then see if the program diverges from its intended goal. White box testing does not account for errors caused by omission, and all visible code must also be readable.

For a complete software examination, both white box and black box tests are required.

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

White Box Testing Techniques:

- **Statement Coverage-** This technique is aimed at exercising all programming statements with minimal tests.
- **Branch Coverage-** This technique is running a series of tests to ensure that all branches are tested at least once.
- **Path Coverage -** This technique corresponds to testing all possible paths which means that each statement and branch is covered.

Calculating Structural Testing Effectiveness:

Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %

Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %

Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %

Advantages of White Box Testing:

- Forces test developer to reason carefully about implementation.
- Reveals errors in "hidden" code.
- Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of White Box Testing:

- Expensive as one has to spend both time and money to perform white box testing.
- Every possibility that few lines of code are missed accidentally.
- In-depth knowledge about the programming language is necessary to perform white box testing.

Test Case:

What is Test case?

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

Typical Test Case Parameters:

- Test Case ID
- Test Scenario

- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters
- Actual Result
- Environment Information
- Comments

Test cases have an important role in protecting your company's reputation. We're all familiar with software products that are released to the market with critical bugs that are later being discovered by the clients. This can be very embarrassing but also can cost your company a great deal of money and even hurt people's lives.

Let me give you two examples:

The infamous iPhone 4 bug was estimated to cost Apple $175 million for free cases in addition to multiple class action lawsuits filed by thousands of disgruntled users.

The 2003 Northeast power blackout was caused by a software alarm system failure. 50 million people lost power for two days; the event caused 11 deaths and cost $6 billion.

These kinds of events can be prevented by using a test cases system.

A test cases system is the backbone of the software testing process and enables you to define a list of tests to make sure every part of your software is properly tested before it is released to the client.

You don't want to leave anything to chance. I'm sure your team is a group of highly trained professionals, but we are all human. And humans sometimes forget things.
You want to make sure that everything has been tested during software testing and nothing has been forgotten.
As a manager, people won't remember how many bugs you have found. Even if you caught a thousand bugs during the testing process, what people will always remember is the one bug that slipped through the cracks and was found by your client.

Element tool enables you to create a closed circuit system that locks the bugs inside and prevents them from slipping through the cracks.

The process of writing test cases is as follows:

You start by creating a test list tree and defining the different features for each requirement. Then you break down each feature into a list of tests. When you're doing this, you want to think about all the ways that people might use each feature and then look for places where it could potentially fail .You create a list of tests for correct behavior that the system should support, to make sure that the product does what it' supposed to do .Then you create a list of tests of incorrect behavior that the system should not support, to make sure the system can handle events that don't follow its business rules.

I'llgiveyouanexample:
Let's say you build a credit card payment page. You want to create a list of tests that check if the page can accept valid credit card details of the different credit card companies. You also should create a list of tests that check if the page rejects invalid credit card details, such as credit cards that have expired, characters instead of credit card numbers, and so on.

Each test needs to have a defined list of steps for the tester to follow in order to complete them. Remember, you don't want to leave anything to chance, and you want to make sure nothing is overlooked. The tester will mark the status of each test as Passed or Failed and then submit the test results into the system. Doing this allows the testing manager to keep track of the progress of the testing.

Make sure to give a priority to each test, running the highest priority tests first before working your way down to the lowest priority. You want to focus on first finding those high priority bugs, the ones that do major damage like crashing the software. If the schedule allows, you can then move on to the lower priority issues.

In the event that a bug is discovered during testing, the tester should submit a new issue to the issue tracking system that describes the bug. The new issue should be assigned to the project manager that needs to add it to the iteration plan and schedule, and then it will be assigned to the appropriate team members for fixing.

9. FUTURE SCOPE & FUTURE ENHANCEMENT

Future enhancements could be done to extend the audio and video abilities of remote desktop sharing. As the tool goes by the open source terms anyone in the world can access the source code and make the changes for the betterment of the System. It will be released as a free service for the use of common man. To ensure the better scalability and performance of the service it will be hosted on Cloud. Mobile based application which will be having the potential to support the service can be developed In this project it is unable to scan any malware and virus in client system and notify it to the server system. Further we want to implement this concept that if any malware or virus found in client systems then a message will be sent to the server system as a notification along with brief details.

Also we want to introduce a concept that, if any pen drive is inserted into client system illegally then automatically it should be formatted and system should shutdown.

CONCLUSION

Remote Desktop Application is one of the best tools in the market for remote administration purposes. It is fast, secure, comfortable to use and platform independent. This enables a desktop administrator to work on any computer or connected to the any system. We can be able to remotely control the server without the aid of a third party product. The existing technologies and additional advanced functions along with a user friendly GUI could create a much faster and simpler remote desktop sharing system to support multiparty conference where in addition to peer-to-peer, more than two users can interact. The project is free and open source, the source code and documentation would be available at the project web site. The multiple clients are handled by providing a time slice to the sockets requesting for data transmission and thus the multiple threads governing the sockets are handled accordingly. The increase in number of users, i.e. the scalability can be taken care of by extending VNC to support multicast data transmission. Whether you want quick access to your home computer from anywhere in the world, remote desktop applications can make your life easier. With the right remote desktop tool, you can access your home computer as though you're sitting right in front of it— no matter where you are, no matter what you're doing.Many remote desktop technologies do not address the **load balancing** and availability problems. Our authentication system provides strong authentication on a **Single Sign On** environment. Our remote desktop architecture addresses conflicts and non interference problems in an efficient manner in terms of computer resources. Our experimentations show that a secured and scalable remote desktop infrastructure can be deployed as an alternative to the current classical desktop models. Our project will reduces the physical strain of the administrator.

BIBLIOGRAPHY

References

www.codeproject.com

www.bdmig33.com

www.google.com