INDORE

Remote PC Connector

A Project Report Submitted at

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal

In partial fulfillment of the degree

Of

Bachelor of Engineering

In

Computer Science & Engineering

Computer Science and Engineering Department
Medicaps Institute of Technology& Management
Indore-453331
2013-14

INDORE

Remote PC Connector

A Project Report Submitted at

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal

In partial fulfillment of the degree

Of

Bachelor of Engineering

In

Computer Science & Engineering

Project Coordinator:	**Submitted By:**
Mr. Arpit Jain	Snehil Vishwakarma (0812CS111105)

Computer Science and Engineering Department
Medicaps Institute of Technology and Management
Indore-453331
2013-14

INDORE

Medicaps Institute of Technology and Management
A.B. Road Pigdambar, Rau, Indore – 453331

CERTIFICATE

This is to certify that **Snehil Vishwakarma (0812CS111105)** has completed his minor project work titled **“Remote PC Connector”** as per the syllabus and submitted a satisfactory report on this project as a part of fulfillment towards the degree of **Bachelor of Engineering (Computer Science)** from Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal.

Signature:

____

Name: _______________

(Project Guide)

Signature:	**Signature:**
_______________	___________________
Name: __________________	**Name:** ____________________
(Internal Examiner)	**(External Examiner)**

ACKNOWLEDGEMENT

The successful completion of a project is generally not an individual effort. It is an outcome of cumulative efforts of a number of persons, each having its own importance to the objective. This section is a value of thanks and gratitude towards every person who has directly or indirectly contributed in their own special way towards the completion of the project. For their invaluable comments and suggestions, we wish to thank them all.

Our first experiment of project has been successful with co-operation of our entire department. We are grateful to our Director **Dr. Sunil Somani**. We are also grateful to our Head of Department **Dr. Pramod Nair** and our class in charge **Ms. Anusha Jain** for her valuable guidance and suggestions throughout the development of project.

No word will suffice to thank our project co-coordinator **Mr. Arpit Jain** for their cooperation, invaluable guidance and encouragement which extended us for completion of our project, both in knowledge and morale.

We also acknowledge all of our team members who dedicated their time and energy for their development efforts.

CONTENTS

Front Page

Certificate

Acknowledgement

Table of Contents

List of figures

List of tables

Abstract

1. Introduction……………………………………………………………………………11

1.1 Aim ……………………………………………………….……………………...12

1.2 Scope………………………………………………………………………….13

1.3 Problems Domain………………………………………………………….......13

1.4 Solution Domain………………………………………………………………14

1.5 Platform Specification…………………………………………………………14

2. System Requirement Analysis……………………………………………………...16

2.1 System Requirement Analysis…………………………………………………...17

2.2 System Feasibility……………………………………………………………..20

3. System Analysis………………………………………………………………………..21

3.1 Use Case Diagram…………………………………………………………….22

3.2 Sequence Diagram…………………………………………………………….24

3.3 Activity Diagram……………………………………………………………….26

4. Implementation……………………………………………………………………..32

4.1 Front Screen……………………………………………………………...33

4.2 Connection Initiation…………………………………………………….34

4.3 Activity Initiation………………………………………………………...36

4.4 Desktop Control Activity………………………………………………...37

4.5 File Sharing Activity……………………………………………………..39

5. Testing……………………………………………………………………………….42

5.1 Testability………………………………………………………………..43

5.2 Testing Method Used…………………………………………………….44

6. Constraints…………………………………………………………………………..48

7. Future Work………………………………………………………………………...49

8. Limitations…………………………………………………………………………..50

9. References…………………………………………………………………………...51

10. Appendices………………………………………………………………………...53

LIST OF FIGURES

Figure no	Figure Name	Page no
3.1.1	Use Case Diagram (Connection Initiation)	23
3.1.2	Use Case Diagram (Desktop Control/File Sharing)	28
3.2.1	Sequence Diagram (Connection Initiation)	25
3.2.2	Sequence Diagram (Desktop Control/File Sharing)	30
3.3.1(a)	Activity Diagram (Connection Initiation – SERVER)	26
3.3.1(b)	Activity Diagram (Connection Initiation – CLIENT)	27
3.3.2	Activity Diagram (Desktop Control/File Sharing)	31
4.1	Front Screen to the Server/Client	33
4.2.1	Server Initiation Phase (Opening port)	34
4.2.2	Initiation Phase (Input of server IP and port number)	35
4.3.1	Completing Connection with Server (Begin Activity at Client side)	36
4.3.2	Activity request by Client (Server side)	36
4.4.1	Client Request for Desktop Sharing of Server screen	37
4.4.3	Desktop Sharing on client screen (Shared server desktop screen)	38
4.5.1	Initiating File/Directory Sharing (Client Side)	39
4.5.2	Selected Directory Sharing (Client Side)	39
4.5.3	Desktop of Client where the folder “Breaking Dawn” is to be copied	40
4.5.4	Directory Sharing Request sent by client to the server (Server Side)	40
4.5.5	“Breaking Dawn” folder copied to Client’s desktop (Client Side)	41
4.5.6	Showing all files are copied to the directory completely (Client Side)	41

LIST OF TABLES

Figure no	Figure Name	Page no
5.2.1	Black Box Testing	45

ABSTRACT

Abstract

What if we want to transfer data from one computer to another, we generally use flash drives, magnetic storage, etc. (in short, EXTERNAL HARDWARE)

But what if there is no external hardware with the users for data transfer between the computers? Moreover, what if the computers are far out of reach at any instance…??

For this the concept of file sharing over computer networks came into play and has solved the problem to a great extent. I.e. Files can be transferred between computers on a computer network without any requirement of travelling from one place to another or having any external hardware to carry the data. External Hardware can also limit the data that can be transferred at once, due to its fixed size, but remote file sharing over network has endless capacity.

With the help of such software, we cannot only reduce the cost and liability on external hardware, but also facilitate access of any computer from any other computer, pre-requisite being both of the computers are active on a working computer network.

INTRODUCTION

1. Introduction

The purpose of this application is to facilitate file transfer between computers on a computer network. It also allows viewing the desktop of the remote computer and controlling it.
The application will be authenticated and it will provide a safe environment for the users.
It may fall short of certain requirements and may require some additional functionality if it is to be used on a real life environment.

Main Features of this application:

- **Desktop Viewing**
- **Desktop Control**
- **Remote File Sharing**
- **Remote Directory Sharing**

1.1 Aim

- To facilitate authenticated remote connection between 2 computers on a computer network.(LAN/MAN/WAN)
- To allow complete desktop control by keyboard and mouse, of one system by another authenticated system.
- To allow heavy data sharing between the 2 connected computers on a network.

1.2 Scope

- This software or application is based on the concepts of networking and various required protocols for file sharing and desktop monitoring over network connections.

- Remote Desktop Control displays the screen of another computer (on any computer network) on your own screen. The program allows you to use your mouse and keyboard to control the other computer remotely.
  It means that you can work on a remote computer, as if you were sitting in front of it, regardless of distance between computers.

- The software also allows you to implement heavy data file transfer (subset of Remote Desktop Sharing) between the computers on network. This restricts control of the user to only file sharing, and not governing the entire remote computer.

1.3 Problem Domain

There were many problems to facilitate file sharing:

1. The basic problem was always of not being able to facilitate unlimited file transfer between computers.
2. And also the distance between the computers were also a major hindrance.
3. We always needed an external hardware, which was not to mention of a fixed size, which was the only way to facilitate file transfer.
4. Even these external drives were not sustainable forever; they even get corrupted due to environmental, usage or handling conditions.

1.4 Solution Domain

This software contains following features to solve the problem specified above:

- Transferring files from a remote computer
- Transferring directories from a remote computer
- Viewing the desktop of the remote computer
- Controlling the desktop of the remote computer

This application allows the user to facilitate unlimited file and directory sharing (prerequisite: active computer network connection on both the computers) which was the main motive of this application.
Additionally the software also facilitates desktop viewing and control of the remote computer it is connected to.

1.5 Platform Specification

1.5.1 Hardware Requirements

- Pentium IV processor, 500 MHz
- Minimum 512 MB RAM
- 20 GB Hard disk
- NIC (Network Interface Card)

1.5.2 Software Requirements

- Windows Operating System.
- Netbeans (Java IDE)
- Configured Network Connections and an active computer Network

Software Technologies Used:

1. GUI (Front End) : JAVA Swings(JDK 1.6)

2. Socket Connections (Back End): JAVA Socket Programming(JDK 1.6)

3. Desktop Sharing : JAVA Robot (JDK 1.6)

4. File Sharing : JAVA Sockets (JDK 1.6)

SYSTEM REQUIREMENT ANALYSIS

2. System Requirement Analysis

In systems engineering and software engineering, requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users.

Systematic requirements analysis is also known as requirements engineering. It is sometimes referred to loosely by names such as requirements gathering, requirements capture, or requirements specification.

The term requirements analysis can also be applied specifically to the analysis proper (as opposed to elicitation or documentation of the requirements, for instance).

Requirements analysis is critical to the success of a development project

Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Conceptually, requirements analysis includes three types of activity:

- **Eliciting requirements:** The task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering.

- **Analyzing requirements:** Determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then resolving these issues.

- **Recording requirements:** Requirements may be documented in various forms, such as natural-language documents, use cases, user stories, or process specifications.

2.1.1 Functional Requirement

User Class 1 - Server

- FR 1.1: Server setup
  This event is to initiate the session on server computer by the user specifying its port number.

- FR 1.2: Permission Grant
  This action takes place when after connection, the client requests for any task (remote desktop control/file sharing/chat messenger). The server can either allow or deny this request.

- FR 1.3: End Activity
  This event can take place anytime by the server, during running of any task between the client and the server.

- FR 1.4: Terminate Connection
  This action can be performed by the sever anytime after a connection has been established. It closes the open port of server.

User Class 2 – Client

- FR 2.1: Client Input
  This event takes place to check the authenticity of the client and establishment of connection with the server. The client is expected to input the correct server IP address and port number.

- FR 2.2: Permission for Remote Activity
  This action is performed by the client after its connection with a server computer. The client can request permission for any activity (remote desktop control/file sharing/chat messenger) to the server.

- FR 2.3: Remote Desktop Control/File Sharing
  After a complete connection with the server, the client can control the server computer's desktop via mouse and keyboard or else could transfer files and folders from server computer to itself.

- FR 2.4: End Activity
  This event can take place anytime by the client, during running of any task between the client and the server.

- FR 2.5: Terminate Connection
  This action can be performed by the client anytime after a connection has been established. It disconnects itself from the port of the server and leaves it open.

2.1.2 Non Functional Requirement

- **Reliability**: By reliability we mean that the application is been simultaneously accessed by a user then no congestion should occur. This is quite reliable because at any instance a computer can only act as a single server and a single client until any of the connection is closed.

- **Availability:** This application is designed for every type of user hence it is free of cost and can be easily available. It is run via java commands to the JVM, so it does not need any software to return. It just needs a platform that is JVM friendly.

- **Maintainability:** The application code is flexible that is whenever some extra things are to be added in future then that can be done easily.

- **Portability:** The application can be run on any computer operating system having JVM installed on it.

2.2 System Feasibility

The developing language is JAVA which is at present the most widely used developing language and in reach with everyone in large variants. Hence this project can reach to wide areas of population and can be used on any OS supporting Java Virtual Machine (JVM).

2.2.1 Operational Feasibility

This application can share any and every kind of file format or directories (random collection of files and folders) over the computer network. It can also facilitate desktop sharing of any operating system irrespective of the server's OS. For instance, it can share an UBUNTU screen on a Windows platform and vice versa.

2.2.2Technical Feasibility

This application is 75-80% efficient in terms of File Sharing/Desktop Control. The signals may sometimes be lost due to congestion or corruption or faults in the streams of sockets over which the data is flowing.

2.2.3 Economical Feasibility

This application requires no extra equipment or software hence is it very economical and cheap.

SYSTEM ANALYSIS

3. System Analysis

Systems analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to requirements analysis or operations research. It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made."

a. Use Case Diagram

In software and systems engineering, a **Use Case** is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human or an external system.

In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in Systems Modeling Language (SysML) or as contractual statements.

3.1.1 Use Case Diagram (Connection Initiation)

Actors:

1) Server: Creates/Opens a socket connection for a client to connect to it and access its files or control its desktop.

2) Client: Connects to the remote server computer to access its files or control its desktop screen.

3) Application System: Manages the server and client connections and the data flowing through the sockets whilst the socket connection.

4) Server System: Acts as a communication medium between the server (user) and the application system.

5) Client System: Acts as a communication medium between the client (user) and the application system.

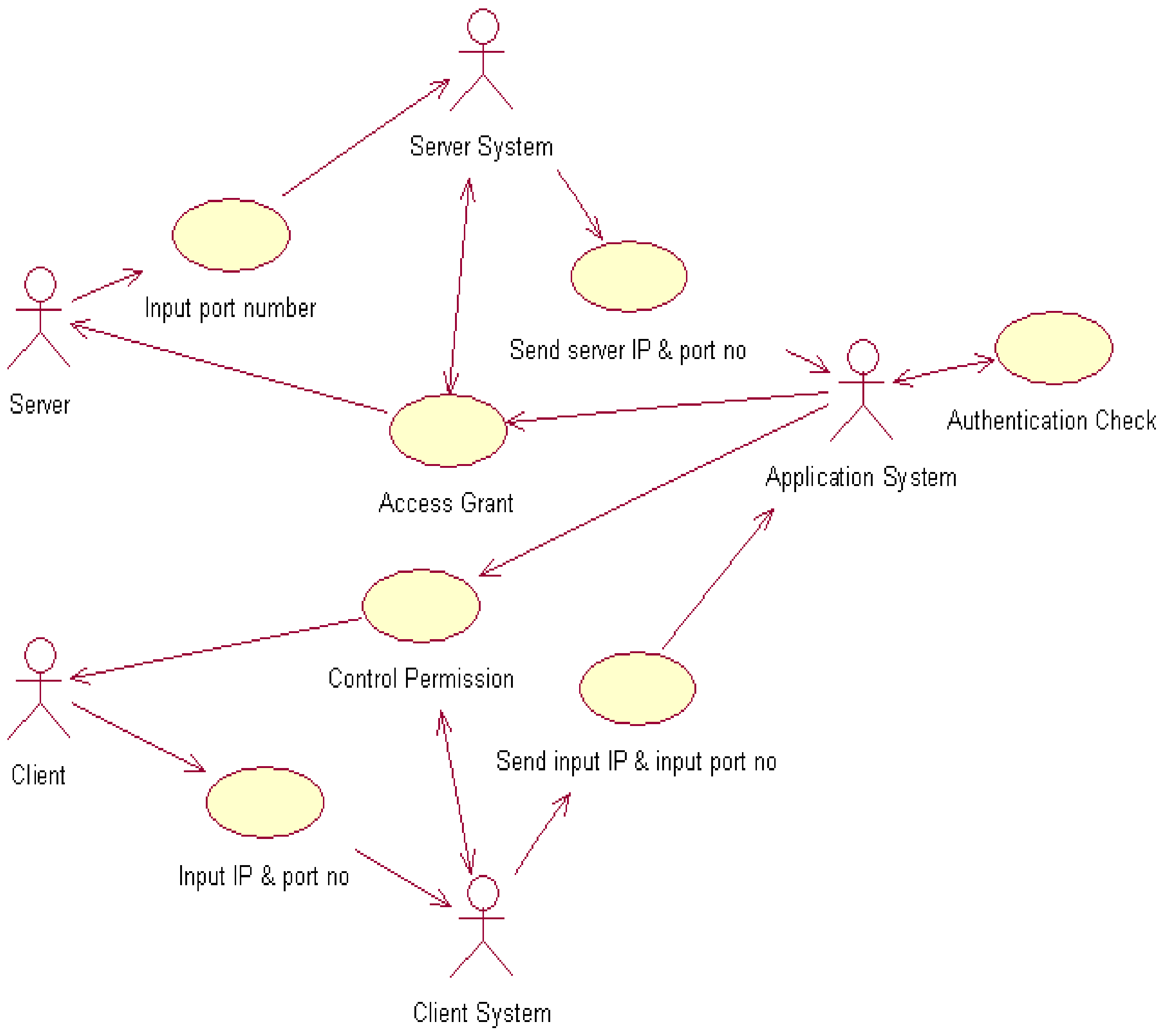


Fig 3.1.1: Use Case Diagram (Connection Initiation)

b. Sequence Diagram

A **sequence diagram** is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios.**

3.2.1 Sequence Diagram (Connection Initiation)

Sequence steps:

Step 1) Server enters the port number to the server system.
Step 2) Server system sends the port number to the application system, which the application system stores.
Step 3) Application system automatically fetches the server's IP address from the server system and stores it.
Step 4) Application System opens a socket connection with the combined server's IP address and port number.
Step 5) Client enters a combination of port number and IP address which it passes to the client system.
Step 6) Client System passes the input IP address and port number by the client, to the application program for authenticity check.
Step 7) If the input combination is authentic, the Application System asks the server to allow the connection or not.
Step 8) If the server allows the connection, the Application System notifies the client user and the connection is completed.

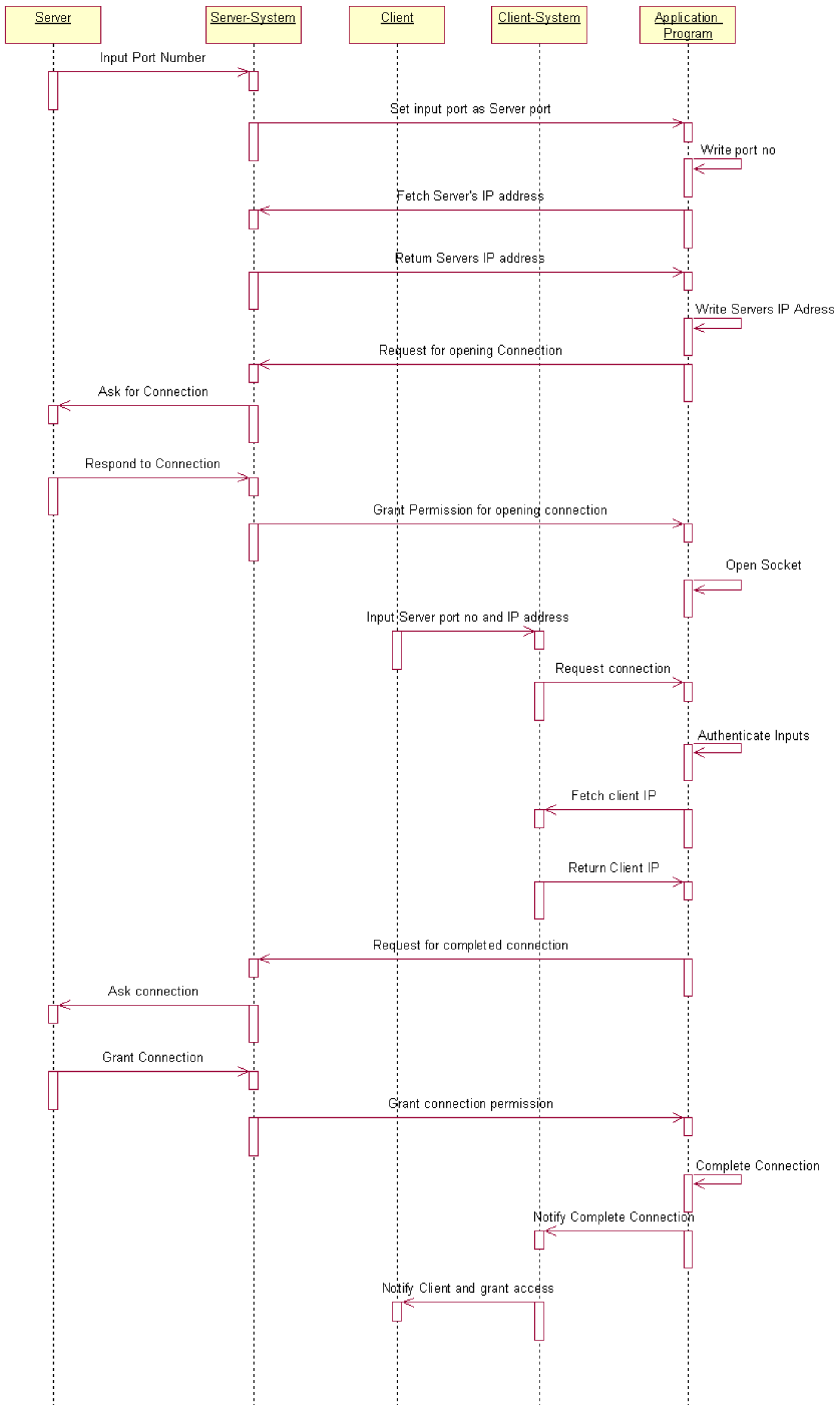

Server
Server-System
Client
Client-System
Application Program
Input Port Number
Set input port as Server port
Write port no
Fetch Server's IP address
Return Servers IP address
Write Servers IP Adress
Request for opening Connection
Ask for Connection
Respond to Connection
Grant Permission for opening connection
Open Socket
Input Server port no and IP address
Request connection
Authenticate Inputs
Fetch client IP
Return Client IP
Request for completed connection
Ask connection
Grant Connection
Grant connection permission
Complete Connection
Notify Complete Connection
Notify Client and grant access

Fig 3.2.1: Sequence Diagram (Connection Initiation)

c. Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows) Activity diagrams show the overall flow of control.

3.3.1 Activity Diagram (Connection Initiation)

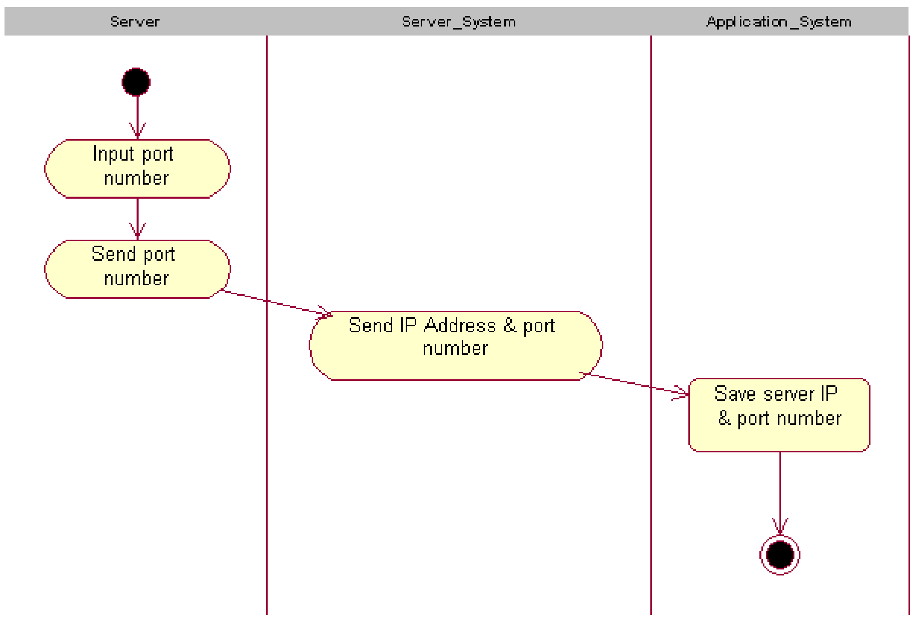


Fig 3.3.1 (a): Activity Diagram (Connection Initiation – SERVER)

The above activity diagram shows how the socket connection is being initiated by the server computer.

- The server enters the port number which is received by the server system.
- The server system also fetches the IP address of the server computer.
- It gives the server IP and port number to the application system, using which the application system creates an open socket connection.

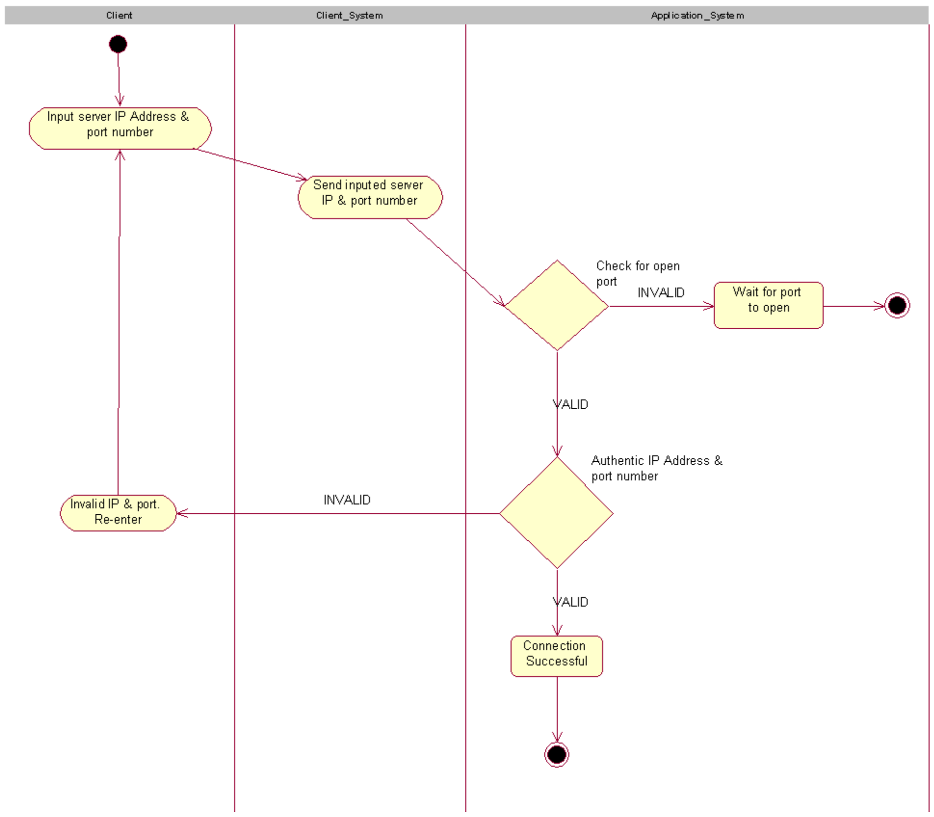

Client
Client_System
Application_System
Input server IP Address & port number
Send inputed server IP & port number
Check for open port
INVALID
Wait for port to open
VALID
Authentic IP Address & port number
INVALID
Invalid IP & port. Re-enter
VALID
Connection Successful

Fig 3.3.1 (b): Activity Diagram (Connection Initiation – CLIENT)

The above activity diagram shows how the socket connection is being completed by the client computer.

- The client enters an IP address and a port number which is received by the client system.
- Client System gives the input IP and port number to the application system for authentication check and completing connection, if an open connection pertains.
- Firstly application program checks for an open port. If yes it checks for authenticated input combination of IP and port no. If no, it declines connection and terminates the client.
- If the input IP address and port number combination is wrong, it asks the client to re-enter the details. If it is right, it notifies the server for connection completion.
- If server allows, the application system completes the socket connection and notifies the client of it and waits for further request by the client.

3.1.2 Use Case Diagram (Desktop Control/File Sharing)

Actors:

6) Server: Allows/Rejects an activity request (file sharing/desktop control) by a client to access its files or control its desktop.

7) Client: Connects to the remote server computer and request permission to access its files or control its desktop screen.

8) Application System: Manages the server and client connections and the data flowing through the sockets whilst the socket connection.

9) Server System: Acts as a communication medium between the server (user) and the application system.

10) Client System: Acts as a communication medium between the client (user) and the application system.

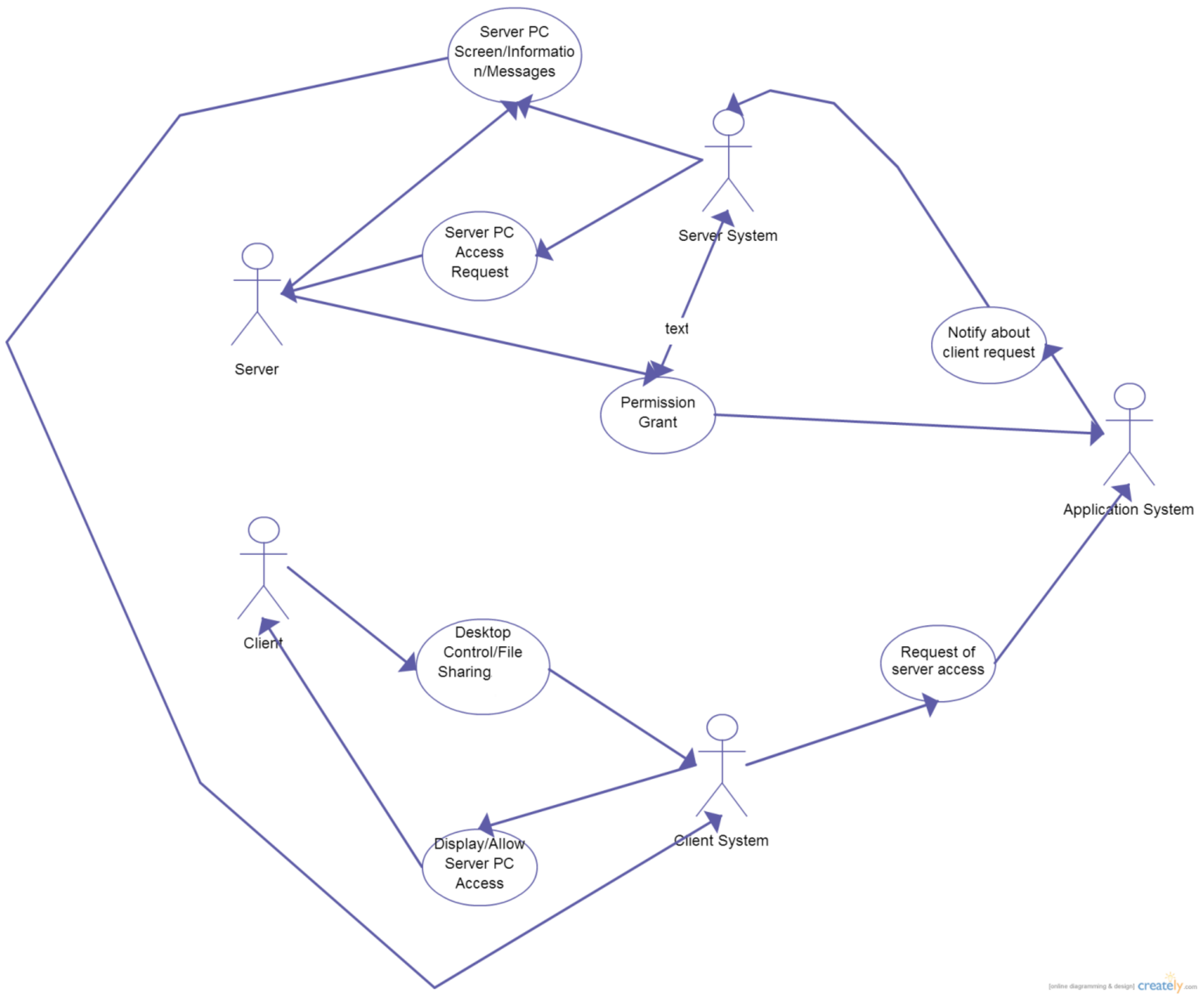


Fig 3.1.2: Use Case Diagram (Desktop Control/File Sharing)

3.2.2 Sequence Diagram (Desktop Control/File Sharing)

Sequence steps:

Step 1) Client sends an activity request (Desktop Control/File Sharing) to the client system.

Step 2) Client system notifies the Application system of the client request.

Step 3) Application system sends this client request to the server via the server system.

Step 4) If the server rejects the request, application system notifies the client via client system. If the server accepts the request, application system notifies the client via client system and waits for further commands by the client.

Step 5) If the client wants to receive files/folders from the server computer, client enters the source path (on the server computer) and the destination path (on the client computer). The transfer of files/folders on the computer is handled by the application system via socket streams.

Step 6) If the client requests for desktop control, the application system continuously grabs server desktop images transmitted by the server system almost seamlessly and also simultaneously hear any keyboard or mouse request for desktop control via the client. The application system handles all these requests during the activity.

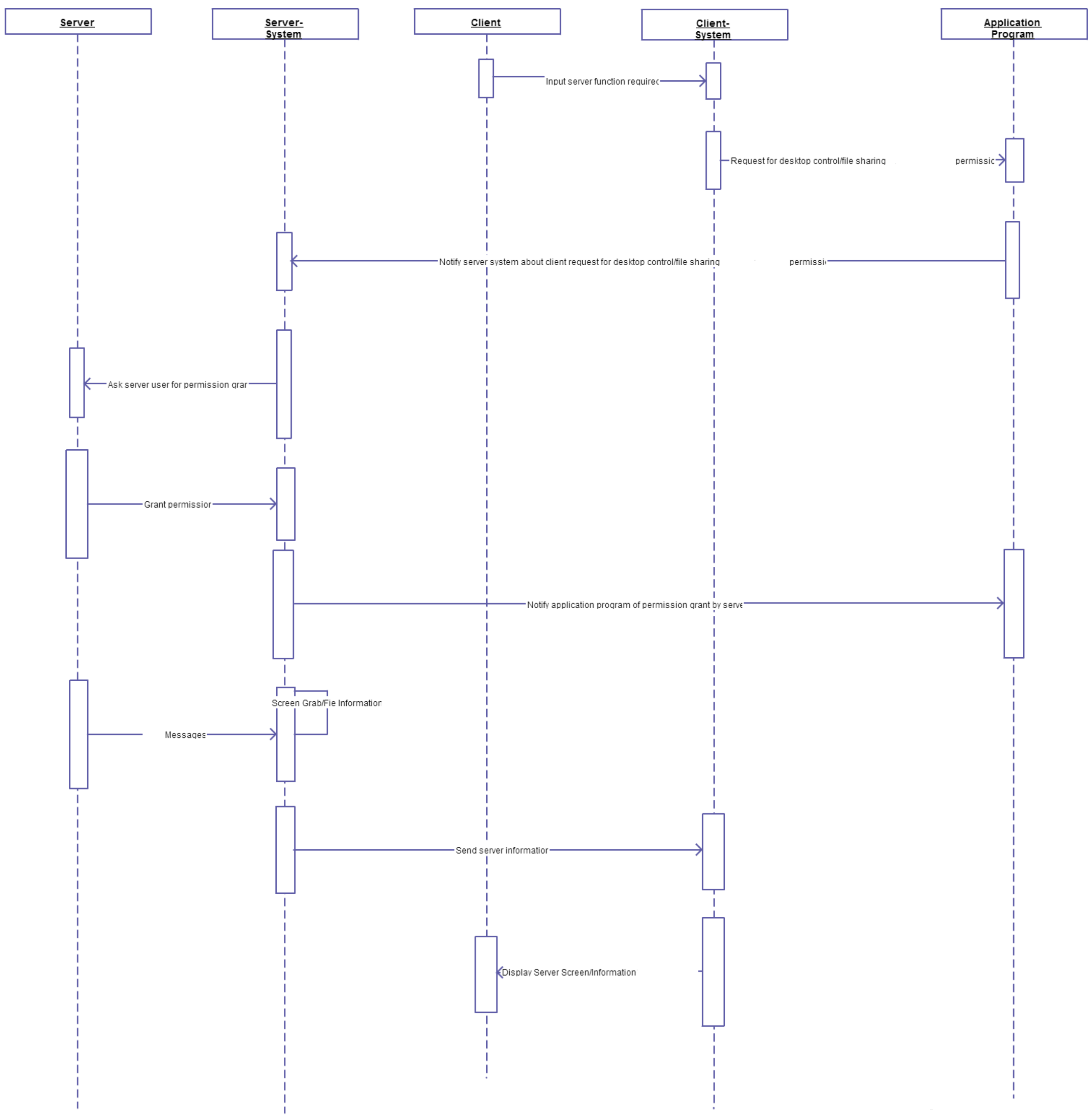
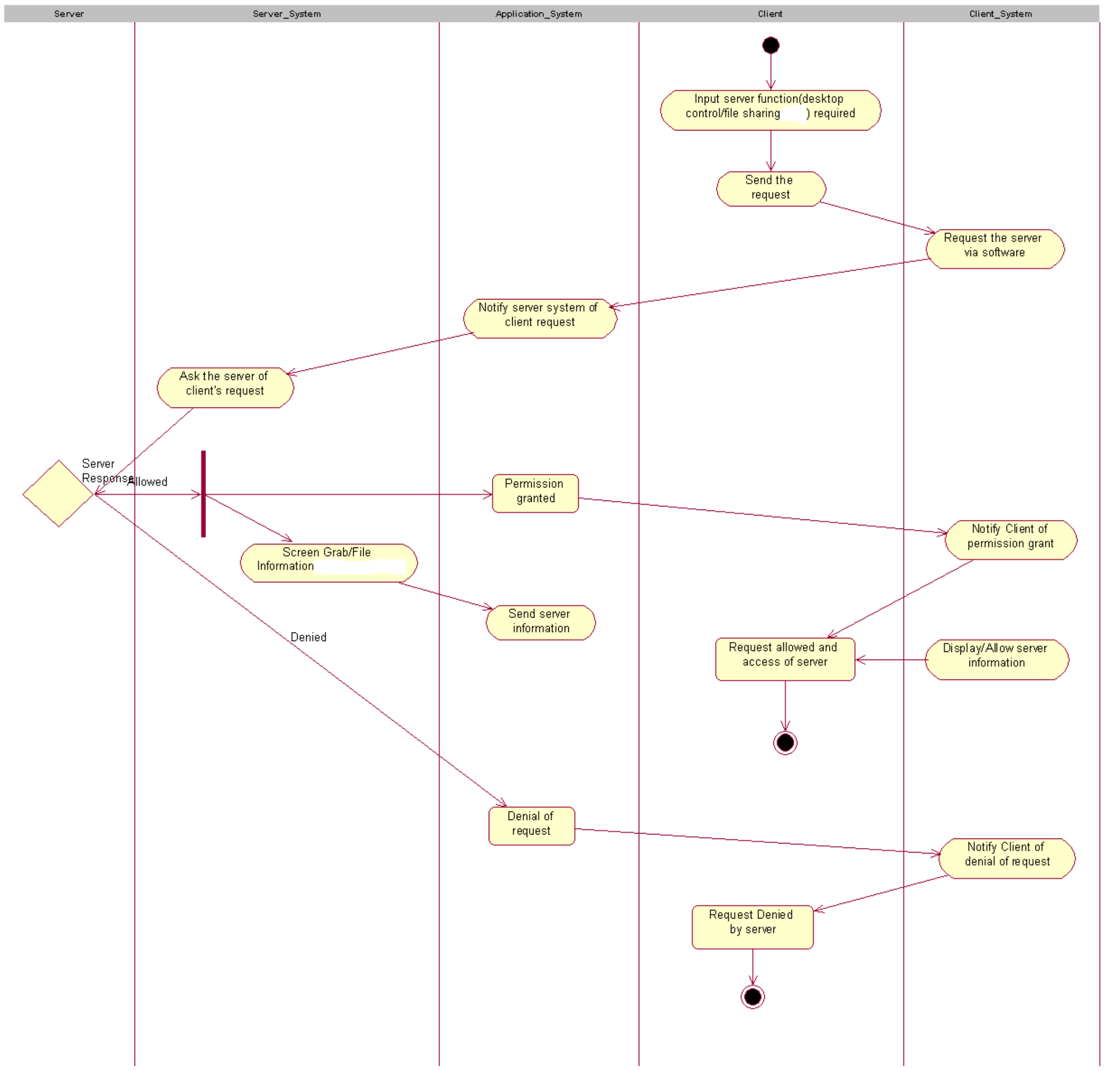

Server
Server-System
Client
Client-System
Application Program
Input server function requirec
Request for desktop control/file sharing
permissic
Notify server system about client request for desktop control/file sharing
permissi
Ask server user for permission grar
Grant permissior
Notify application program of permission grant by serve
Screen Grab/Fie Informatior
Messages
Send server informatior
Display Server Screen/Information

Fig 3.2.2: Sequence Diagram (Desktop Control/File Sharing)

3.3.2 Activity Diagram (Desktop Control/File Sharing)



Server
Server_System
Application_System
Client
Client_System
Input server function(desktop control/file sharing) required
Send the request
Request the server via software
Notify server system of client request
Ask the server of client's request
Server Response
Allowed
Denied
Permission granted
Notify Client of permission grant
Screen Grab/File Information
Send server information
Request allowed and access of server
Display/Allow server information
Denial of request
Notify Client of denial of request
Request Denied by server

Fig 3.3.2: Activity Diagram (Desktop Control/File Sharing)

The above activity diagram shows how an activity is carried out between the client and the server computer.

- The client requests for an activity (desktop control/file sharing) to the server.
- The server may accept/reject the client request.
- If the permission is granted by the server, the file transfer or the desktop control commands are processed and maintained by the application system.

IMPLEMENTATION

4. Implementation

4.1 Front Screen

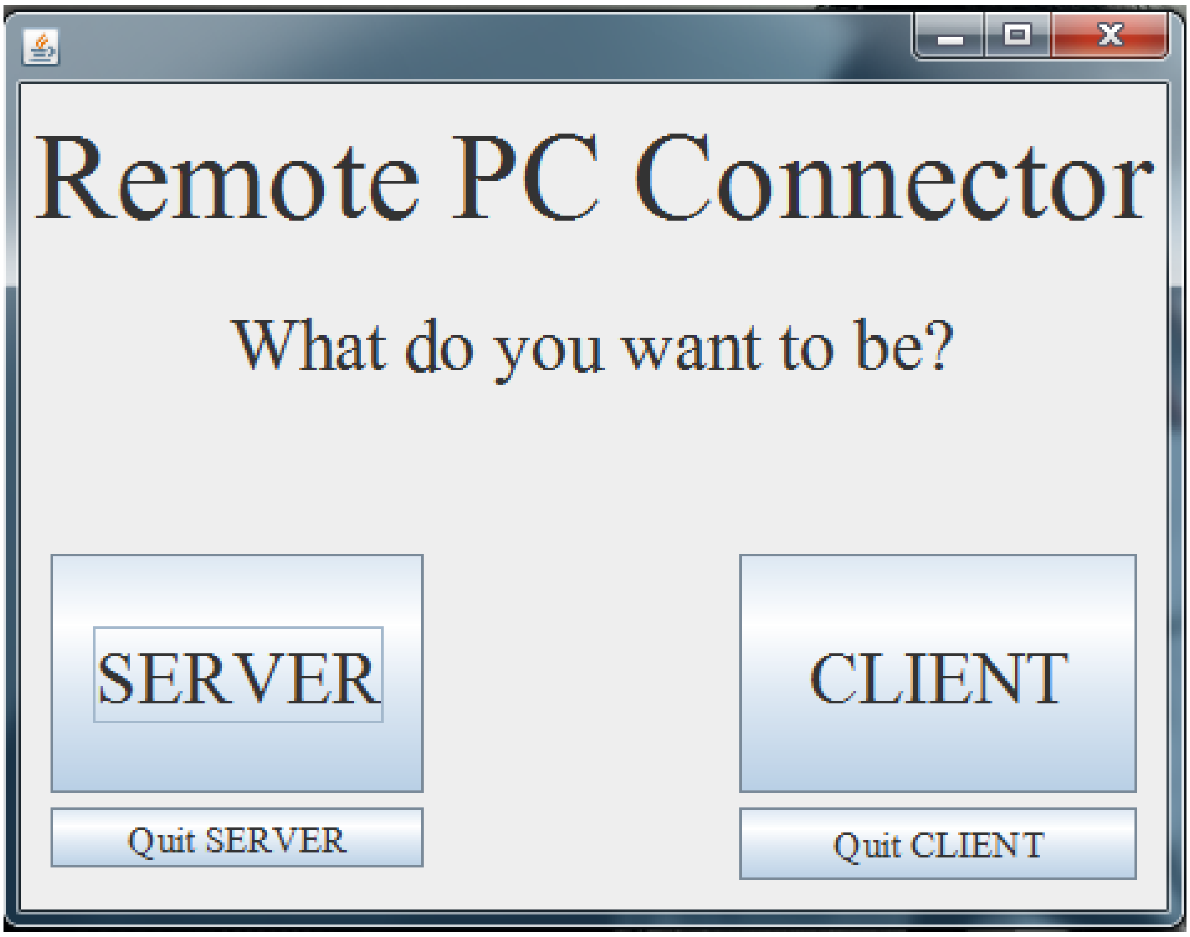


Fig 4.1: Front Screen to the Server/Client

Description:

This image shows the main screen of the application. It gives the user to be either the server computer (the one sharing its data), or the client computer (the one accessing the server's data).

It also allows both, the server and the client to end the connection at any point of time after the connection has been established.

4.2 Connection Initiation

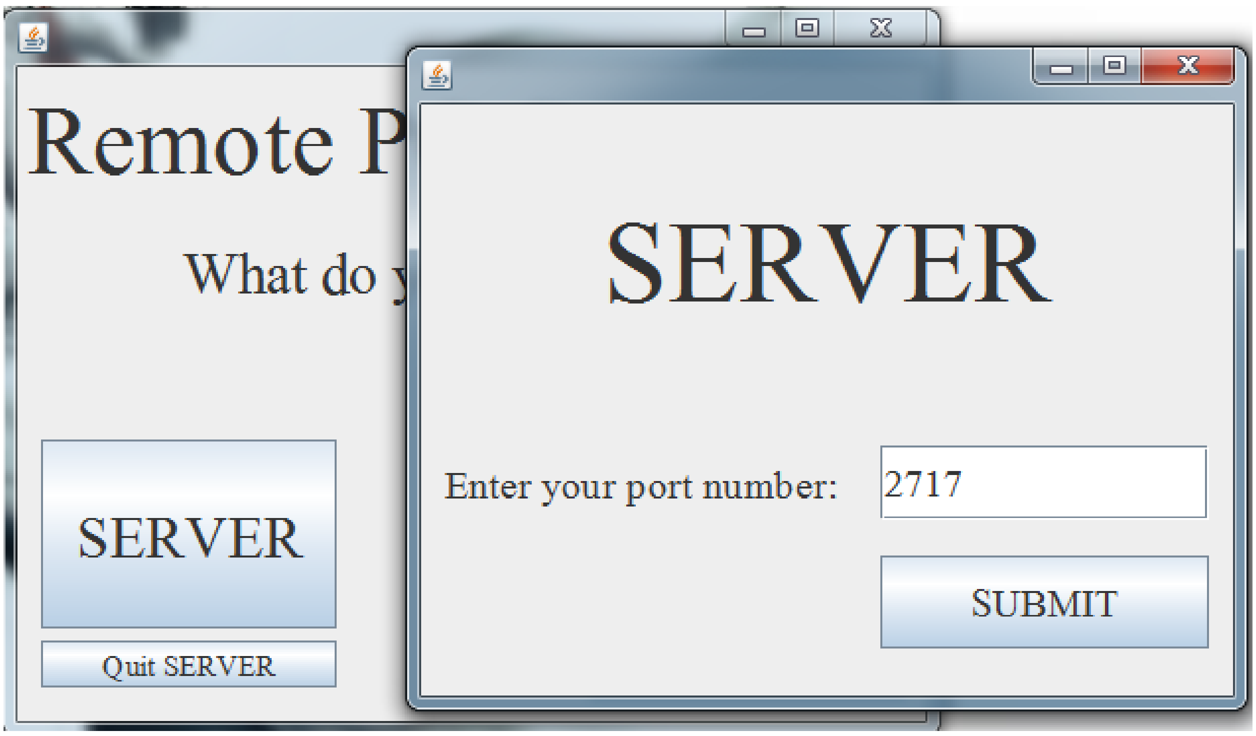


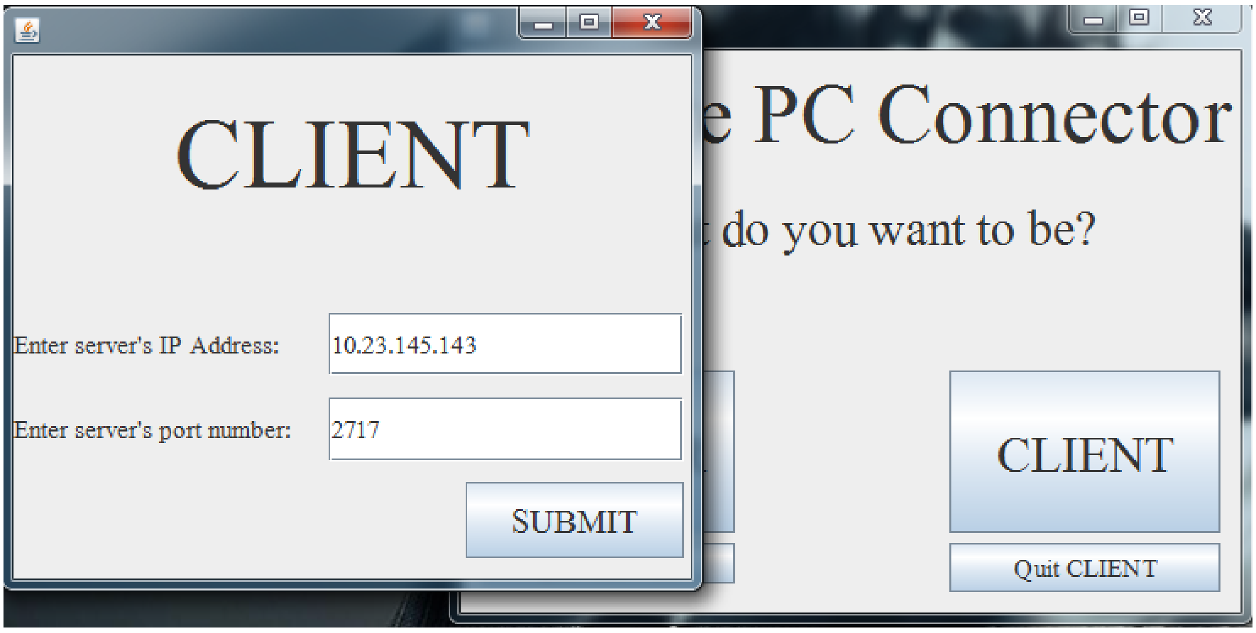
Fig 4.2.1: Server Initiation Phase (Opening port)

Description:

The user is trying to open a server connection, and thus it has to input a port number for the process.

This port number with the combination with the user's IP address (automatically fetched) opens a socket connection and waits for connection request by a client computer.

The port number to be entered by the server user should be greater than 1024, since port number up to 1024 are reserved for various Internet protocols.



Enter server's IP Address: 10.23.145.143

Enter server's port number: 2717

Fig 4.2.2: Client Initiation Phase (Input of server IP and port number)

Description:

The user is trying to complete connection, and thus it has to input the correct combination of server IP address and chosen port number for the process.

The input IP address and port number combination goes under authentic check. If the check is successful, the socket connection is completed and the client computer is asked for the activity it wants to perform. If not, the client is asked to re-enter the IP address and port number.

4.3 Activity Initiation

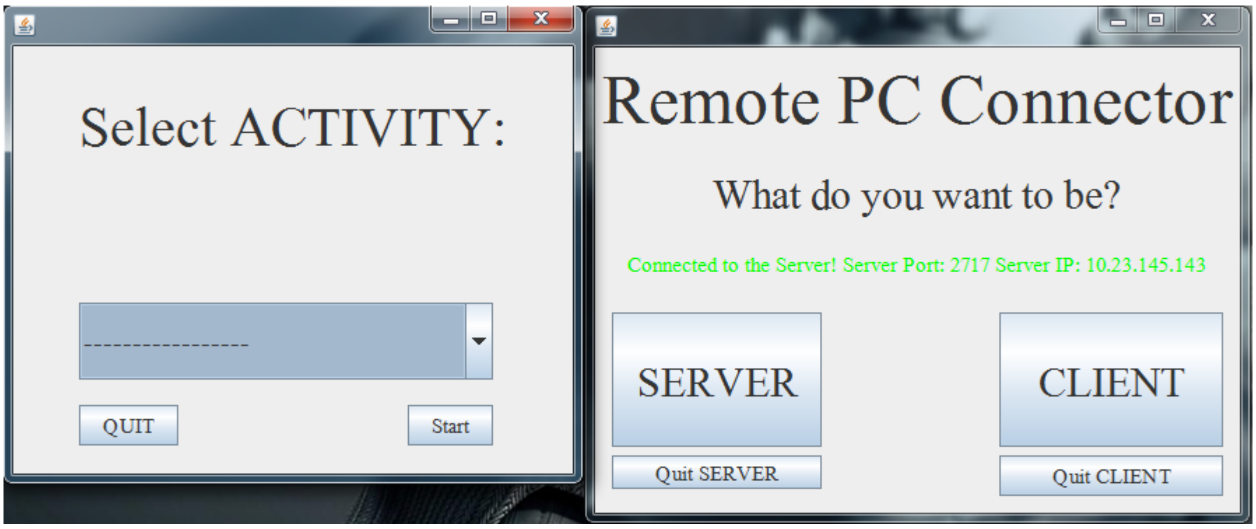


Fig 4.3.1: Completing Connection with Server (Begin Activity at CLIENT SIDE)

Description:

The client computer is asked for the activity (desktop control/file sharing) it wants to perform. Once it selects an activity, an activity request is sent to the server.

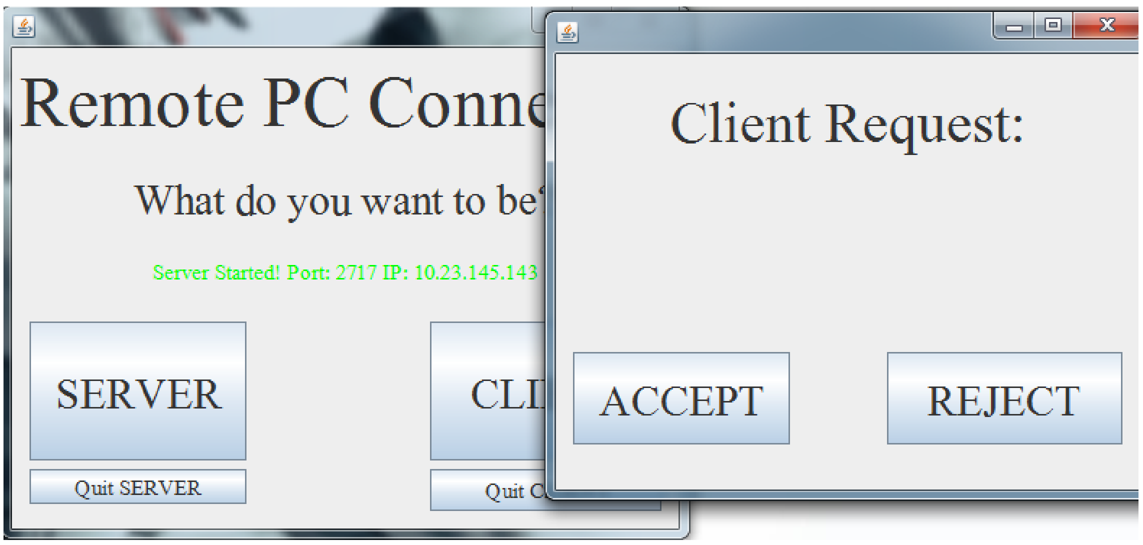


Fig 4.3.2: Activity request by Client (SERVER SIDE)

Description:

The activity request is received by the server. It can either accept or reject the request.

4.4 Desktop Control Activity

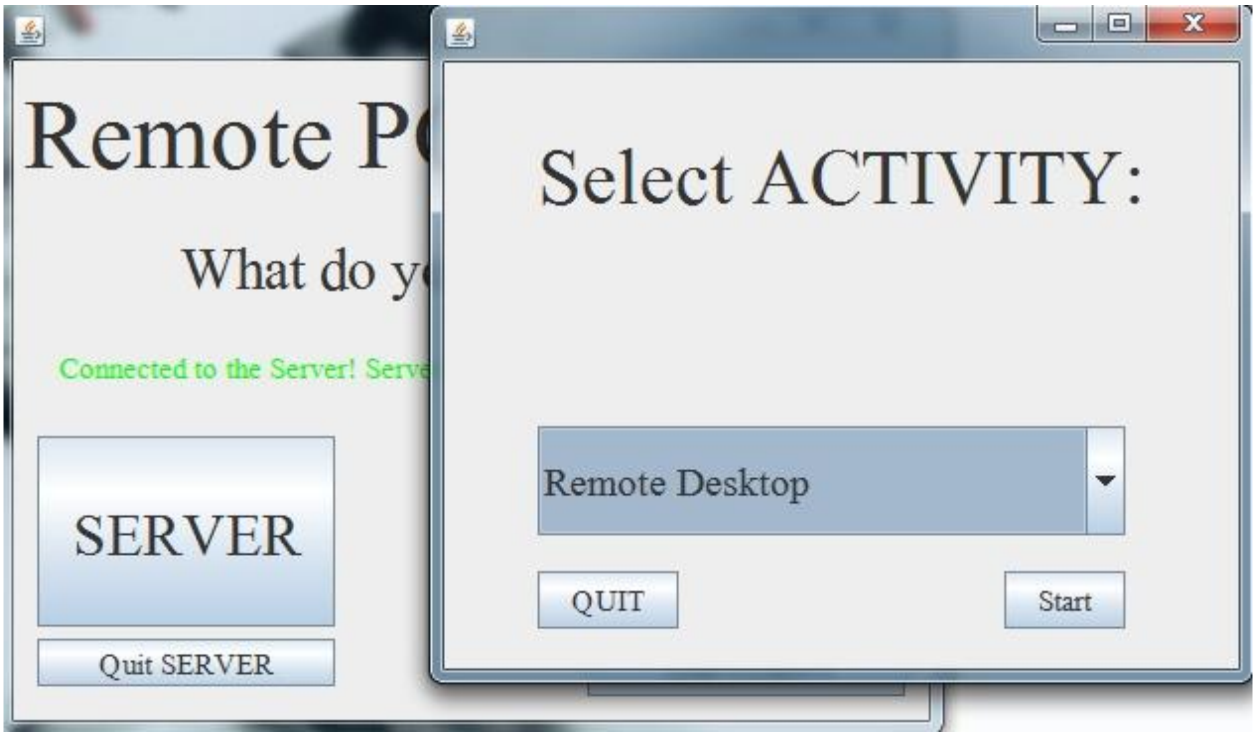
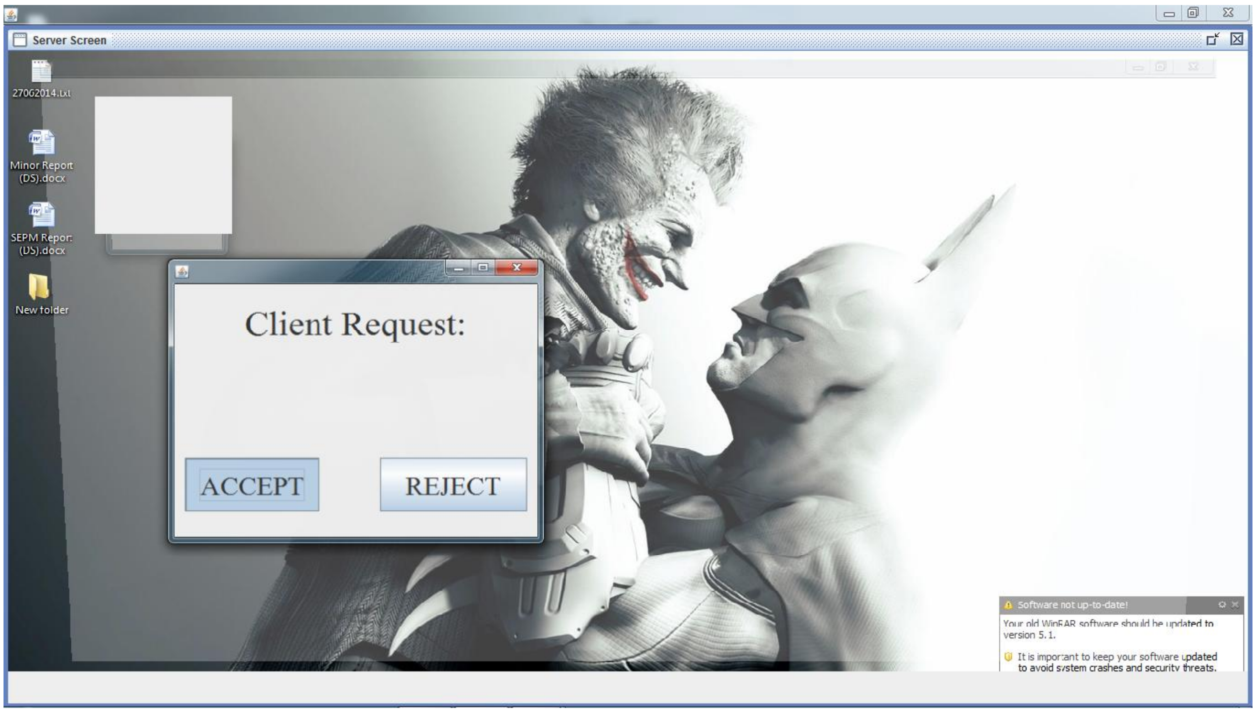


Fig 4.4.1: Client Request for Desktop Sharing of Server screen

Description:

The client request for “Remote Desktop” Activity. The request is sent to the server (Refer Fig. 4.3.2)



Software not up-to-date!
Your old WinRAR software should be updated to version 5.1.
It is important to keep your software updated to avoid system crashes and security threats.

Fig. 4.4.2: Desktop Sharing on client screen (Shared server desktop screen)

Description:

The server desktop screen visible on the client computer. The client can also control the server computer’s keyboard and mouse from its local keyboard and mouse.

4.5 File Sharing Activity

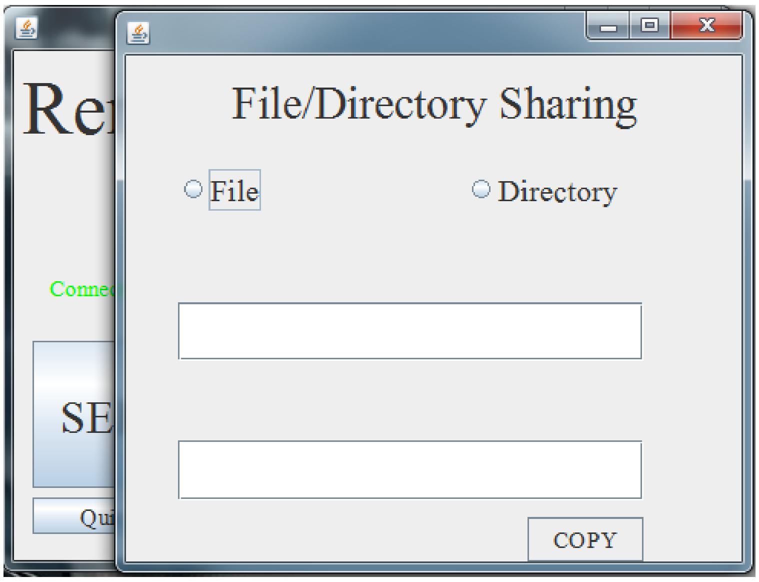


Fig 4.5.1: Initiating File/Directory Sharing (Client Side)

Description:

File sharing initiated by client. It can either request for a single file or a complete directory.

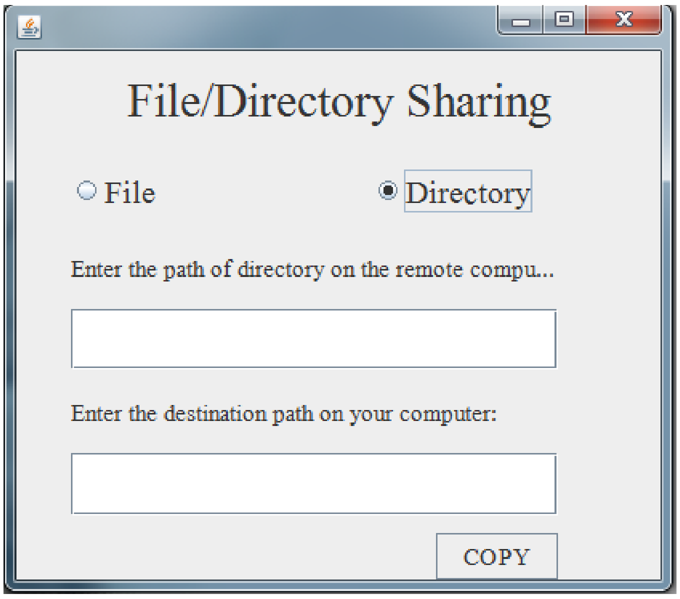
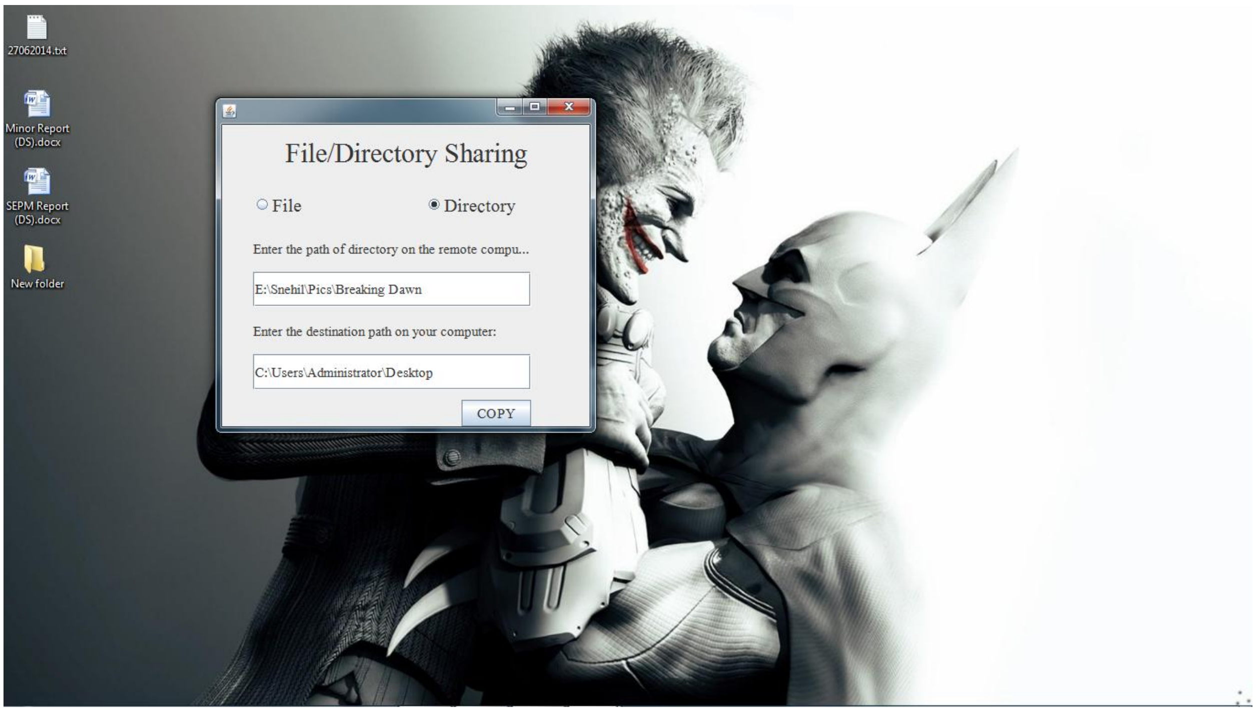

File/Directory Sharing
File
Directory
Enter the path of directory on the remote compu...
Enter the destination path on your computer:
COPY

Fig 4.5.2: Selected Directory Sharing (Client Side)

Description:

The client selects directory sharing. It has to specify the path of the directory (to be copied) on the server computer and the path on local computer where the directory is to be saved.



File/Directory Sharing

File [✓ Directory]

Enter the path of directory on the remote compu...

E:\Snehil\Pics\Breaking Dawn

Enter the destination path on your computer:

C:\Users\Administrator\Desktop

Fig 4.5.3: Desktop of Client where the folder "Breaking Dawn" is to be copied

Description:

Showing desktop screen where the folder is to be copied (before the file transfer).

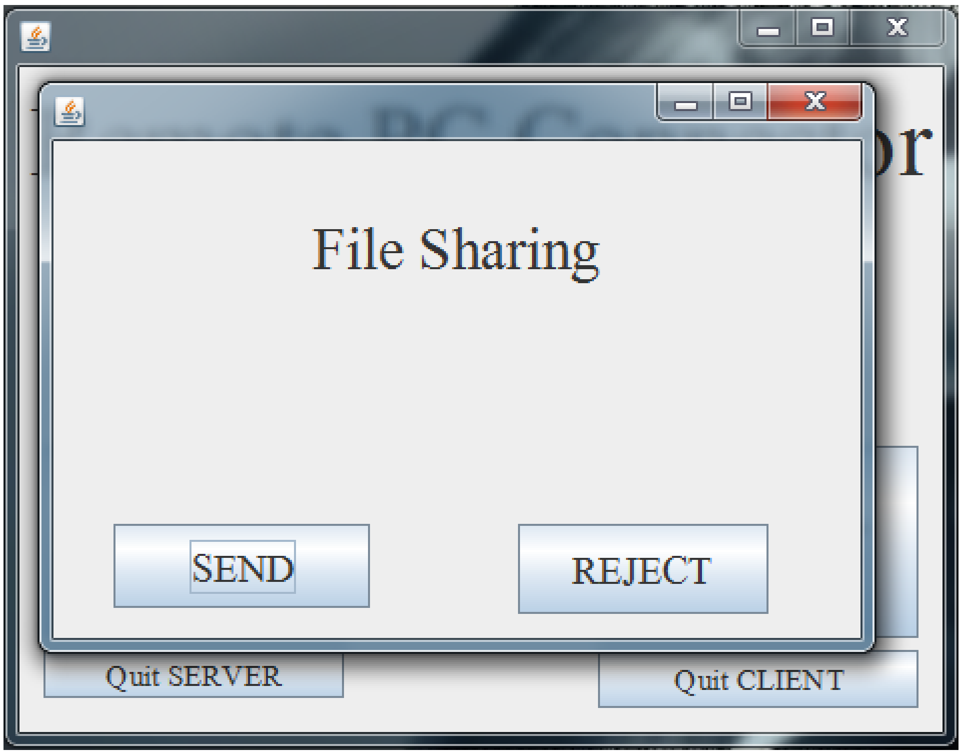


Fig 4.5.4: Directory Sharing Request sent by client to the server (Server Side)

Description:

File sharing request is sent to the server. It can either allow the file transfer or deny.

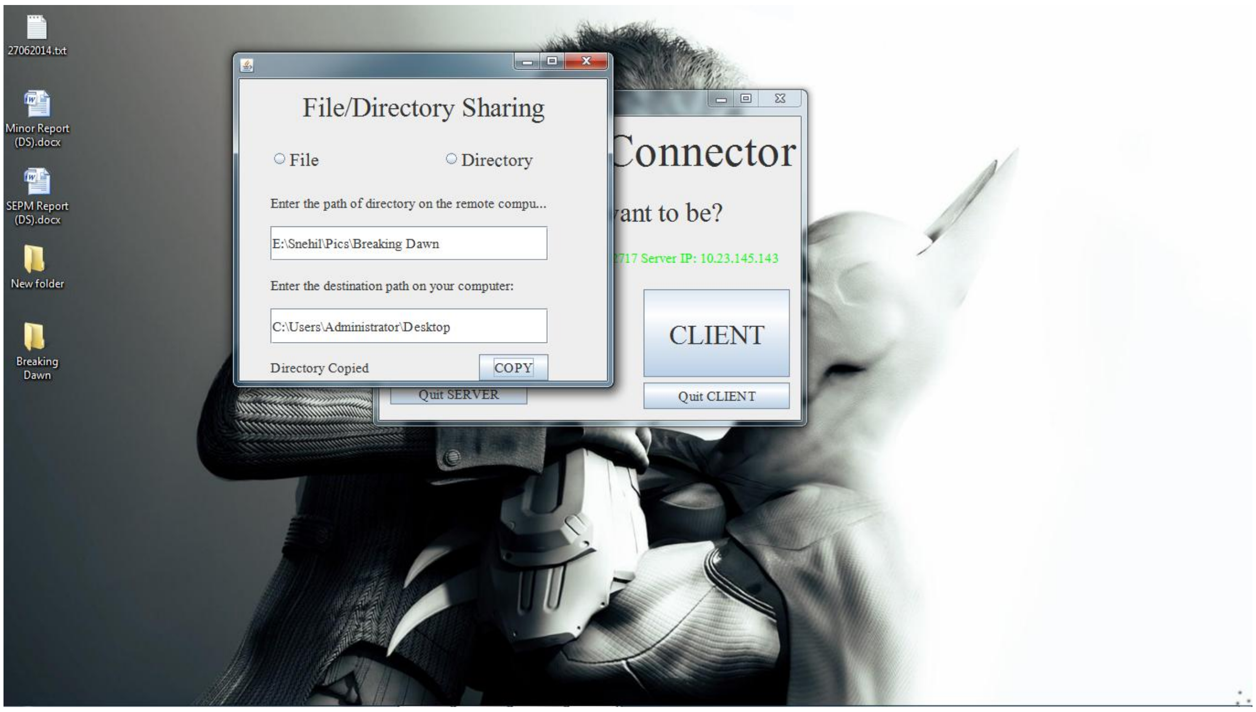

27062014.txt
Minor Report (DS).docx
SEPM Report (DS).docx
New folder
Breaking Dawn
File/Directory Sharing
File
Directory
Enter the path of directory on the remote compu...
E:\Snehil\Pics\Breaking Dawn
Enter the destination path on your computer:
C:\Users\Administrator\Desktop
Directory Copied
COPY
Quit SERVER
Connector
ant to be?
717 Server IP: 10.23.145.143
CLIENT
Quit CLIENT

Fig 4.5.5: "Breaking Dawn" folder copied to Client's desktop (Client Side)

Description:

Showing desktop screen where the folder is copied (after the file transfer).

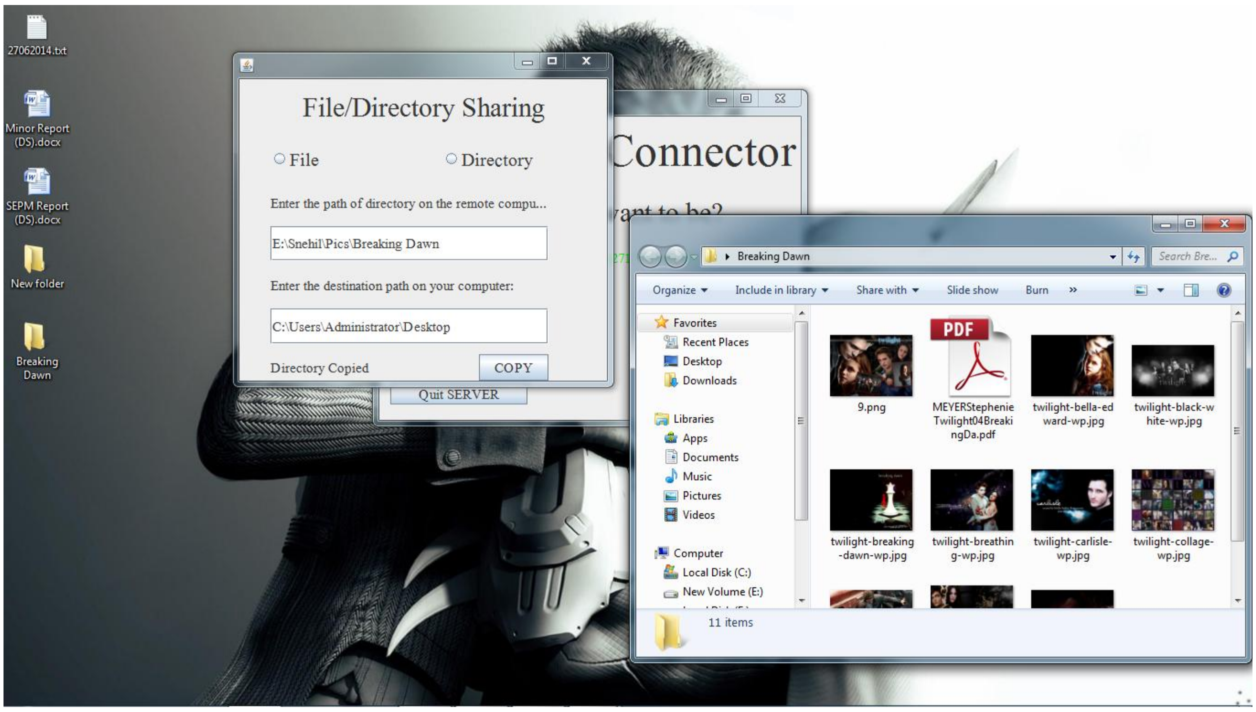


Fig 4.5.6: Showing all files are copied to the directory completely (Client Side)

Description:

Showing the folder open, all the files of the folder in the server computer are copied.

TESTING

5. Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.

Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

Test techniques include, but are not limited to the process of executing a program or application with the intent of finding Software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- meets the requirements that guided its design and development,
- works as expected,
- can be implemented with the same characteristics,
- Satisfies the needs of stakeholders.

5.1 Testability

Software testability is the degree to which a software artifact (i.e. a software system, software module, requirements- or design document) supports testing in a given test context.

Testability is not an intrinsic property of a software artifact and cannot be measured directly (such as software size). Instead testability is an extrinsic property which results from interdependency of the software to be tested and the test goals, test methods used, and test resources (i.e., the test context).

A lower degree of testability results in increase test effort. In extreme cases a lack of testability may hinder testing parts of the software or software requirement sat all.

5.1.1 Observability

Observability is defined as the degree to which the results of an innovation are visible to others. The easier it is for individuals to see the results of an innovation, the more likely they are to adopt it. Such visibility stimulates peer discussion of a new idea, as friends and neighbors of an adopter often request innovation-evaluation information about it.

5.1.2 Decomposability

The design method provides a systematic mechanism for decomposing the problem into sub problems, which will reduce the complexity of the overall problem thereby achieving an effective modular solution. The project is decomposed into certain modules on which further testing are applied.

5.1.3 Stability

The software stability is defined as the ability of the software to withstand abrupt errors or unknown condition. The software must be able to deal with all the changes and maintain the stability.

5.2 Testing Method Used

Any engineered product can be tested in one of two ways:

White-Box Testing:

Knowing the internal workings of a product, tests can be conducted to ensure that the internal operation performs according to specification and all internal components have been adequately exercised.

Black-Box Testing:

Black box testing, also called “Behavioral testing”, focuses on the functional Requirements of the software. It enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. It is a complementary approach to “White-Box testing” that is likely to uncover a different class of errors.

Black Box testing attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures
- Behavior or performance errors
- Initialization and termination errors.

Test No	Operation	Input	Expected Output	Actual Output	Result
1.	Input port number (server)	Server port number	Open port at server	Open port at server	Open connection
2.	Input server IP & port number (client)	Server IP address & Server port number	Complete connection with server	I. Wrong IP & Correct port	Reject connection
				II. Correct IP & Wrong port	Reject connection
				III. Wrong IP & Wrong port	Reject connection
				IV. Correct IP & Correct port	Complete connection

Test No	Operation	Input	Expected Output	Actual Output	Result
3.	Request Activity (Client)	Choice of Activity	Allow activity	Depends on server	Server response
4.	Response to request (Server)	I. Allow II. Reject	-	-	I. Permission grant I. Permission denial
5.	End Activity (Server/ Client)	Button click on End activity	End Activity & return to the choice menu	End Activity & return to the choice menu	End of Activity
6.	Terminate Connection (Server/ Client)	Button click on terminate connection	Termination of connection	Termination of connection	End of connection between client and server

Table No. 5.2.1 Black Box Testing

Limitations, Future Scope and Conclusion

6. Limitations to the project

- As our software is network based, both the host computer and remote computer must be running and connected to the same computer network while working on.

- The remote computer must always be running the software so as the user to remotely access it.

- As of now the software is only been tested on the Windows operating system. (XP or higher)

- The software works only on computers (Desktops and Laptops), not on any other handheld device.

- The software also does not support multiple client-server connections, and thus no broadcast file sharing or desktop screening is possible which is a major drawback.

- The software cannot run on its own on the server computer as it was desired. It was done to provide secured file sharing or desktop control even after the connection is established.

7. Future scope to the project

- The software can establish to control and share files and folders simultaneously, which it can do but separately.
- The software can and should provide higher levels of security and authentication checks.
- The software can have many other additional useful features like chat messenger, screenshots, video-shots, voice and video messenger, etc. that can increase the functionality of remote access.
- Also available on handheld devices.
- Can also facilitate multiple client-server connection, thus providing broadcast file sharing and desktop screening.

8. Conclusion

The aim of the project was to connect remote computers and share information and data, reducing cost and time for data sharing.

We have successfully established connection, but only between 2 computers on a network, and allow desktop control or heavy file sharing between the computers.

This software is just a small step into the power of computer networks. It has endless scope and future advancement possibilities.

REFERENCES

9. References

Books referred

The following books were used extensively for the project development and implementation.

1. "The Complete Reference Java2" Tata McGraw-Hill publishing Company Limited
- Herbert Schildt.
2. "Programming With Java: A Primer 3E" Tata McGraw-Hill publishing Company Limited
   - E. Balagurusamy

Websites Referred

- http://stackoverflow.com
- http://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html
- http://forum.codecall.net/topic/54172-robot-class

APPENDICES

10. Appendices

Net Beans:

Net Beans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5.It is also an application platform framework for Java desktop applications and others.

The Net Beans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The Net Beans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the Net Beans Platform (including the Net Beans IDE itself) can be extended by third party developers.

Sockets:

A network socket is an endpoint of an inter-process communication flow across a computer network. Today, most communication between computers is based on the Internet Protocol; therefore most network sockets are Internet sockets.

A socket API is an application programming interface (API), usually provided by the operating system, that allows application programs to control and use network sockets. Internet socket APIs are usually based on the Berkeley sockets standard.

A socket address is the combination of an IP address and a port number, much like one end of a telephone connection is the combination of a phone number and a particular extension. Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

An Internet socket is characterized by a unique combination of the following:

- Local socket address: Local IP address and port number
- Remote socket address: Only for established TCP sockets. As discussed in the client-server section below, this is necessary since a TCP server may serve several clients concurrently. The server creates one socket for each client, and these sockets share the same local socket address from the point of view of the TCP server.
- Protocol: A transport protocol (e.g., TCP, UDP, raw IP, or others). TCP port 53 and UDP port 53 are consequently different, distinct sockets.

Within the operating system and the application that created a socket, a socket is referred to by a unique integer value called a socket descriptor. The operating system forwards the payload of incoming IP packets to the corresponding application by extracting the socket address information from the IP and transport protocol headers and stripping the headers from the application data.

Robot Class:

This class is used to generate native system input events for the purposes of test automation, self-running demos, and other applications where control of the mouse and keyboard is needed. The primary purpose of Robot is to facilitate automated testing of Java platform implementations.

Using the class to generate input events differs from posting events to the AWT event queue or AWT components in that the events are generated in the platform's native input queue. For example, Robot.mouseMove will actually move the mouse cursor instead of just generating mouse move events.

Note that some platforms require special privileges or extensions to access low-level input control. If the current platform configuration does not allow input control, an AWTException will be thrown when trying to construct Robot objects.