A

REPORT

ON

REMOTE DESKTOP

By

DeepanshuSoni (101350)

Under the guidance of

Dr. Nitin

Department of Computer Science & Engineering

Jaypee University of Information Technology

Waknaghat, Solan

HP - 173234

CERTIFICATE

This is to certify that the work titled “Remote Desktop Sharing”, in partial fulfillment for the award of degree of B.Tech of Jaypee University of Information Technology, Waknaghat, Himachal Pradesh has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor: ……

Name of Supervisor: Dr. Nitin

Designation: Senior Lecturer

Date:

ACKNOWLEDGEMENTS

We would like to express our great gratitude towards our supervisor Dr**. Nitin** who has given us support and suggestions. Without his help we could not have presented this dissertation up to the present standards. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at REMOTE DESKTOP.

LIST OF FIGURES AND TABLES

CONTENTS

ABSTRACT

The project "**Remote Desktop Sharing**" is basically aimed to provide communication between server and client from the local area network for capture client screen to server.

Current thin-client remote desktop systems were designed for data-oriented applications over low-quality LAN links and they do not provide satisfactory end-user performance in enterprise environment for more and more popular graphical and multimedia applications. To improve perception of those applications in thin-client environment we designed architecture of a server-side Quality of Service (QoS) management component responsible for mapping application QoS requirements into network QoS. We analyze how service differentiation and traffic management techniques combined with user perception monitoring can be used in order to adjust network level resource allocation when performance of multimedia applications in remote desktop environment is not meeting user requirements. Our objective is to provide QoS-aware remote desktop systems which will be able to manage available resources in intelligent manner and meet end-user performance expectations.

Chapter One
INTRODUCTION

Introduction

1.1 Overview

Remote Desktop Sharing is designed and developed with the end user in mind, user in a similar way. The features included in the Remote Monitoring package are a jump further on of other similar fields.

A Remote Desktop Sharing is a desktop application designed for use on a control client pc by server. A Remote Desktop is optimized so as to display client pc which is connected in local area network. Remoter Desktop software must be small and efficient to accommodate the low memory capacity.

1.2 Purpose

This system is useful for those offices which have multiple users in different locations and admin want to monitor all users in an office. This Application also provides server monitor all clients in an office. In this, server is able to chat with client and also can chat with client and server is also able to shut down client pc. These facilities are provided in which place where local area network is available.

1.3 Modules of the project

Remote Server

This is the server part which waits for clients connections and per each connected client, a new frame appears showing the current client screen. When you move the mouse over the frame, this results in moving the mouse at the client side. The same happens when you right/left click mouse button or type a key while the frame is in focus.

Remote Client

This the client side, its core function is sending a screen shot of the client's desktop every predefined amount of time. Also it receives server commands such as "move the mouse command", then executes the command at the client's PC.

Coding Structure

a. Remote Server

ServerInitiator Class

This is the entry class which listens to server port and waits for client’s 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 Internal Frameper 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.

b. RemoteClient

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.

Chapter Two

Requirement Gathering

2.1 Primary Research

User requirement is the main point of view for any developmental movement. User accepts only those products which are able to complete their functions. Primary research is very supportive in receiving information from user. Developer did the primary research in order to verify the feasibility of the proposed system. The following activities were conducted in order to gain information from the users.

Sites and Books:

1. Bellinaso Marco (2006). ASP.NET 2.0 Website Programming: Problem - Design - Solution (Programmer to Programmer). Paperback
2. Matthew MacDonald, Matthew MacDonald, and Julian Templeman (2005). Beginning ASP.NET 2.0 in C# 2005: From Novice to Professional
3. Shahram Khosravi. ASP.NET AJAX Programmer's Reference: with ASP.NET 2.0 or ASP.NET 3.5 (Programmer's Reference (Wrox))
4. Schwable, Kathy, Information Technology Project Management, Cengage Technology, 2008

2.2 Questionnaires

A questionnaire is basically a survey. It is usually a short survey that takes specific information. This kind of fact-finding method is applied to obtain more information from the people who are significant for the system and having very fewer times for participa

tion for the reason that of their daily schedule. The questionnaires are also important for congregation information from the users who are related to the project but are geographically separated from each other.

The primary intention of distributing of this questionnaire is to further justify and to gain the user support of implementation of the proposed system and to know the kind of feature that those respondents anticipate and as well as any existing system that the user came across. A total of 15 respondents have taken part this survey. The questions in the questionnaires are as follows:

1. Do you want scheduler in the system?

☐ Yes

☐ No

Justification: If you are too often send messages on the schedule, the Email Scheduler plug in will help you fully automate this process. Now you can set a certain date and time to send out specific messages. Furthermore, Email scheduler can work with attached a file, which allows you to, for example, send certain documents periodically.

2. Are you using any current system?

☐ Yes

☐ No

Justification: This question would help me to know that whether users want or not to use such kind of system.

3. Do you want user groups?

☐ Yes

☐ No

Justification: When you are in an Active Directory network environment, you can set Smart mail policies to enforce settings on a specific or a group of users. This is mainly

to be used to change or limit the default behavior of Outlook in a corporate environment but can also be useful in some home environments. For instance, as a home user you might want to set policies on what your children can and cannot do in Outlook.

4. Do you want attaches from one group to another?

☐ Yes

☐ No

Justification: this would ask to know, whether the user want to use this functionality or not.

5. Do you face any problem in current system?

☐ Yes

☐ No

6. **Do you want your customize setting or just system default?**

☐ Yes

☐ No

Justification- Designed to find out is user want customized setting or system default, which cannot be change by user.

2.3 Technical Research

What is a Methodology?

Software engineering is carry out of using preferred procedure techniques to progress the quality of a software development effort. A methodology is defined as a collection of procedures, techniques, tools, and documentation aids which will help developers in their efforts (both product and process related activities) to implement a new system. For successful implementation, a well-organized and systematic approach is crucial.

Therefore, several methodologies were developed to encourage the systematic approach to planning, analysis, design, testing and implementation. Methodologies offer various tools and techniques to assist in analysis, design and testing in terms of detailed design of software, data flowcharts and database design.

Why Methodology?

1. To complete a project within time and budget with the expected scope and quality we need methodologies which provide for a framework.
2. Most methodologies have a general planning, developing and managing stages in common. They suggest the development team the ways of thinking, learning and arriving at a regular feasible solution.

To select an ideal methodology was based on project requirements and goals.

- Functional Decomposition: The methodology should have stages according to the interrelated activities which can be grouped into different functional areas.
- Requirement Changes: If required, methodology provides scope to change the requirement.
- Manage Risks: Determined the risk is an important activity to develop a project.
- Iterative approach: Iteration allows refinement of requirement as well as design.
- Documentation: Methodology provides support for large documentation.
- Analysis and Design Support: A well defined structure of the methodology helps for analysis and designing to development process..
- Implementation: The system should be implemented as per plan.
- Testing Support: More testing, more reliable the product is.

Object Oriented Approach: Object oriented concepts will be used in developing the project as it supports component reusability.

2.4 Selection of Programming language:

To successfully develop a project or system, technical and programming skills are both equally important. The academic research determines the design of the system, while the technical and programming research will determine the usability of the system. The objective of this session is to identify a programming language platform for developing this project. Most important factors such as productivity, maintainability, efficiency, portability, etc. pay an enormous part in this track. As the project being developed is an email application to be developed by an object oriented approach, it leaves the developer to choose from the following languages: JAVA, ASP.NET and C#. A lot of research was carried to select the best among these.

2.4.1 Java:

It is an object oriented and platform independent language which is used for programming desktop application. It consists of a virtual machine and set of libraries which are needed to allow the use of file systems, networks, graphical interfaces, etc.

Java is a programming language originally developed by Sun Microsystems and released in 1995 a core component of Sun Microsoft, java platform. The language derives much of its syntax from C and C++ but a simpler object model and fewer low level facilities. Java applications are typically component to byte code that can run on any java virtual machine (JVM) regardless of computer architecture. The original and reference implementation of java compliers, virtual machines, and class libraries which is developed by Sun from 1996, as of May 2007, in compliance with the specifications of the Java community Process, sun made available most of their Java technologies as free software under the General public license. Others have also developed alternatives implementation of technologies such as GNU complier of Java and GNU Classpath. Java is a programming language originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a

general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java is currently one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Classpath.

James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time. The language was initially called Oak after an oak tree that stood outside Gosling's office; it went by the name Green later, and was later renamed Java, from Java coffee, said to be consumed in large quantities by the language's creators. Gosling aimed to implement a virtual machine and a language that had a familiar C/C++ style of notation.

Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "Write Once, Run Anywhere" (WORA), providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network-and file-access restrictions. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular. With the advent of Java 2 (released initially as J2SE 1.2 in December 1998–1999), new versions had multiple configurations built for different types of platforms. For example, J2EE targeted enterprise applications and the greatly stripped-down version J2ME for mobile applications (Mobile Java). J2SE designated the Standard Edition. In 2006, for marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, and Java SE, respectively.

In 1997, Sun Microsystems approached the ISO/IEC JTC1 standards body and later the

Ecma International to formalize Java, but it soon withdrew from the process. Java remains a de facto standard, controlled through the Java Community Process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System. Sun distinguishes between its Software Development Kit (SDK) and Runtime Environment (JRE) (a subset of the SDK); the primary distinction involves the JRE's lack of the compiler, utility programs, and header files.

On November 13, 2006, Sun released much of Java as free and open source software, (FOSS), under the terms of the GNU General Public License (GPL). On May 8, 2007, Sun finished the process, making all of Java's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.

Sun's vice-president Rich Green said that Sun's ideal role with regards to Java was as an "evangelist". Following Oracle Corporation's acquisition of Sun Microsystems in 2009–2010, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency". Java software runs on laptops to data centres, game consoles to scientific supercomputers. There are 930 million Java Runtime Environment downloads each year and 3 billion mobile phones run Java. On April 2, 2010, James Gosling resigned from Oracle.

2.4.2 Swings:

Swing is the primary Java GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar

components such as buttons, check box and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

Since early versions of Java, a portion of the Abstract Window Toolkit (AWT) has provided platform-independent APIs for user interface components. In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system.

By contrast, Swing components are often described as lightweight because they do not require allocation of native resources in the operating system's windowing toolkit. The AWT components are referred to as heavyweight components.

Much of the Swing API is generally a complementary extension of the AWT rather than a direct replacement. In fact, every Swing lightweight interface ultimately exists within an AWT heavyweight component because all of the top-level components in Swing (JApplet, JDialog, JFrame, and JWindow) extend an AWT top-level container. Prior to Java 6 Update 10, the use of both lightweight and heavyweight components within the same window was generally discouraged due to Z-order incompatibilities. However, later versions of Java have fixed these issues, and both Swing and AWT components can now be used in one GUI without Z-order issues.

The core rendering functionality used by Swing to draw its lightweight components is provided by Java 2D, another part of JFC.

2.4.3 Relationship to SWT

The Standard Widget Toolkit (SWT) is a competing toolkit originally developed by IBM and now maintained by the Eclipse community. SWT's implementation has more in common with the heavyweight components of AWT. This confers benefits such as more accurate fidelity with the underlying native windowing toolkit, at the cost of an increased exposure to the native platform in the programming model.

The advent of SWT has given rise to a great deal of division among Java desktop developers, with many strongly favouring either SWT or Swing.

There has been significant debate and speculation about the performance of SWT versus Swing; some hinted that SWT's heavy dependence on JNI would make it slower when the GUI component and Java need to communicate data, but faster at rendering when the data model has been loaded into the GUI, but this has not been confirmed either way. A fairly thorough set of benchmarks in 2005 concluded that neither Swing nor SWT clearly outperformed the other in the general case.

SWT is considered by some to be less effective as a technology for cross-platform development. By using the high-level features of each native windowing toolkit, they claim that SWT returns to the issues seen in the mid-1990s (with toolkits like zApp,Zinc, XVT and IBM/Smalltalk) where toolkits attempted to mask differences in focus behaviour, event triggering and graphical layout. Failure to match behaviour on each platform can cause subtle but difficult-to-resolve bugs that impact user interaction and the appearance of the GUI.

2.4.4 Java features

The fundamental forces that necessitated the invention of Java are portability and security. There are other factors that played an important role in modeling the final form of the language. The Java team is as follows added up the key considerations and features.

Java was designed to be easy for professional programmer to learn and use effectively. Java is completely object-oriented so if we are sell versed with OOP'S learning Java another attribute that makes it easy to learn it makes an effort not have surprising Features. In Java, there are a small number of clearly defined ways to accomplish a given task.

Although influenced by its predecessor, Java was not designed to be source code compatible with any other languages. This allowed Java team the freedom to design with a blank slate. One out come of this was a clean usable programmatic approach to objects. Borrowing liberally from any seminal object software environment of the last few decades. Java manages to strike a balance purist's ―everything is an object‖ paradigm, and the pragmatist's ―stay out of my way‖ model. The object model in Java

is simple and easy to extent, while simple types, such as integers are kept as high performance non-objects.

1. Java is Portable:

One of the biggest advantages Java offers is that it is portable. An application written in Java will run on all the major platforms. Any computer with a Java based browser can run the applications or applets written in the Java programming language. A programmer no longer has to write one program to run on a Macintosh, another program to run on a Windows machine, still another to run on a Unix machine and so on. In other words, with Java, developers write their programs only once.

The virtual machine is what gives Java a cross platform capabilities. Rather than being complied into machine language, which is different for each operating systems and computer architecture, Java code is compiled into byte codes. With other languages, the program code is complied into a language that the computer can understand. The problem is that other computers with different machine instruction set cannot understand that language. Java code, on the other hand is complied into byte codes rather than a machine language. These byte codes go to the Java virtual machine, which executes them directly or translates them into the language that is understood by the machine running it.

In summary, these means that with the JDBC API extending Java, a programmer writing Java code can access all the major relational databases on any platform that supports the Java virtual machine.

2. Java is Object – Oriented:

Java is Object Oriented, which makes program design focus on what you are dealing with rather than on how you are going to do something. This makes it more useful for programming in sophisticated projects because one can break the things down into understandable components. A big benefit is that these components can then be reused.

Object oriented languages use the paradigm of classes. In simplest term, a class includes both the data and the functions to operate on the data. You can create an instance of a

class, also called an object, which will have all the data members and functionality of its class. Because of this, you can think of a class as being like template, with each object being a specific instance of a particular type of class.

The class paradigm allows one to encapsulate data so that specific data values are those using the data cannot see function implementation. Encapsulation makes it possible to make the changes in code without breaking other programs that use that code. If for example the implementation of a function is changed, the change is invisible to another programmer who invokes that function, and it does not affect his/her program, except hopefully to improve it.

Java includes inheritance, or that ability to derive new classes from existing classes. The derived class, also called subclass, inherits all the data and the function of the existing class, referred to as the parent class. A subclass can add new data members to those inherited form the parent class. As far as methods are concerned, the subclass can reuse the inherited methods, as it is, or change them, or even add its own new methods.

3. Java Makes It Easy:

In addition to being portable and object oriented, Java facilitates writing correct code. Programmers spend less time writing Java code and a lot less time debugging it. In fact, developers have reported slashing development time by as much as two thirds.

Java automatically takes care of allocating and the reallocating memory, a huge potential source of errors. If an object is no longer being used (has no reference to it), then it is automatically removed from memory, or Garbage Collected by a low priority daemon thread called Garbage Collector. Java's no pointer support eliminates big source errors. By using object references instead of memory pointers, problems with pointer arithmetic are eliminated, and problems with inadvertently accessing the wrong memory address are greatly reduced.

Java's strong typing cuts down on runtime errors, because Java enforces strong type checking, many errors are caught when code is complied. Dynamic binding is possible and often very useful, but static binding with strict type checking is used when possible.

Java keeps code simple by having just one way to do something instead of having several alternatives, as in some languages. Java also stays lean by not including multiple inheritances, which eliminates the errors and ambiguity that arise when you create a subclass that inherits from two or more classes. To replace capabilities, multiple inheritances provide, Java lets you add functionality to a class throw the use of interfaces.

4. Java is Extensible:

A big plus for Java is the fact it can be extended. It was purposely written to be lean with the emphasis on doing what it does very well, instead of tying to do everything from the beginning; it was return so that extending it is very easy. The java platform includes an extensive class library so that programmers can use already existing classes, as it is, create subclasses to modify existing classes, or implement to augment the capabilities of classes.

5. Java is Secure:

It is important that a programmer not be able to write subversive code for applications or applets. This is especially true with the Internet being used more and more extensively for services such as electronic commerce and electronic distribution of software and multimedia content. The way memory is allocated and laid out. In java an object's location in memory is not determined until the runtime, as opposed to C and C++. As the result, a programmer cannot look at a class definition and figure out how it might be laid out in memory. Also since, Java has no pointers; a programmer cannot forge pointers to memory. The Java Virtual Machine (JVM) doesn't trust any incoming code and subjects it to what is called Byte Code Verification. The byte code verifier, part if the virtual machine, checks that

The format of incoming code is correct

- Incoming code doesn't forge pointers.
- It doesn't violate access restrictions.
- It access objects as what they are

The Java byte code loader, another part of the JVM, checks whether classes loaded during program execution are local of from across a network. Imported classes cannot be substituted for built in classes, and built in classes cannot accidentally reference classes bring in over a network.

The Java Security manager allows user to restrict entrusted Java applets so that they cannot access the local network, local files and other resources.

6. Java Performs Well:

Java performance is better than one might expect. Java's many advantages, such as having built in security and being interpreted as well as complied, do have a cost attached to them. However, various optimizations have been built, in, and the byte code interpreter can run very fast the cost it doesn't to do any checking. As a result, Java has done quite respectably in performance tests. Its performance numbers for interpreted byte codes are usually more than adequate to run interactive graphical end user applications. For situations that require unusually high performance, byte codes can be translated on the fly generating the final machine code for the particular CPU on which the application is running at run time. Java offers good performance with the advantages of high-level languages but without the disadvantages of C and C++. In the world of design trade-off, you can think of Java as providing a very attractive middle ground.

7. Java is Robust:

The multiplatform environment of the WEB places extraordinary demands on a program, because it must execute reliably in a variety of systems. Thus the ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in a few key areas to force you to find your mistakes early in program developments. At the same time, Java frees you from having to worry about many of the most common causes of programming errors. Because Java is strictly typed language, it checks your code at compile time. However, it also checks your code at run time. In fact, many hard to track down bugs that often turn up in hard to reproduce runtime situations are simply impossible to create in Java. Knowing that what you have

written will behave in a predictable way under diverse conditions is a key feature of Java.

8. Java is Multithreaded:

Multithreading is simply the ability of a program to do more than one thing at a time. For example an application could be faxing a document at the same time it is printing another document. Or a program could process new inventory figures while it maintains a feed for current prices. Multithreading is particularly important in multimedia: a multimedia program might often be running a movie, running a audio track and display in text all at the same time.

2.4.5 The Java Platform

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other, hardware-based platforms.

The Java platform has two components:

1.The Java Virtual Machine (Java VM)

2.The Java Application Programming Interface (Java API)

Java VM is the base for the Java platform and is ported onto various hardware-based platforms.

Programs written in the Java programming language are first compiled and then interpreted.

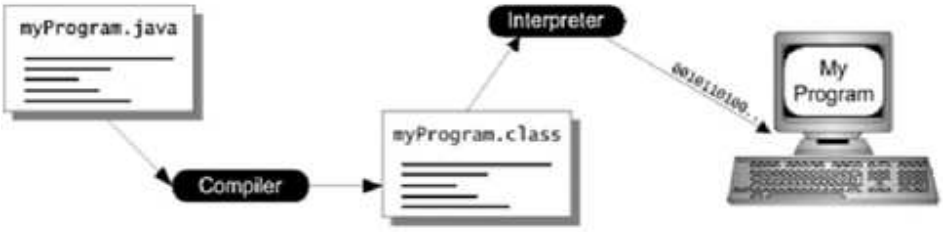


FIG 2.1 Program written in the Java programming language

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section highlights what functionality some of the packages in the Java API provide.

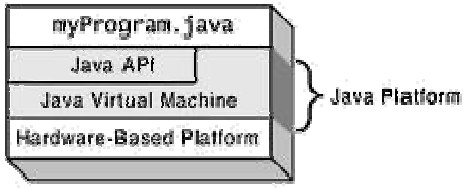


FIG 2.2 The Java API and the Java VM insulate a program from hardware dependencies.

.2.4.6 Implementation of Java Technology

Every full implementation of the Java platform gives you the following features:

1. The essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

2. Applets: The set of conventions used by Java applets.

3. Networking: URLs, TCP (Transmission Control Protocol), UDP (User Datagram Protocol) sockets, and IP (Internet Protocol) addresses.

4. Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

5. Security: Both low level and high level, including electronic signatures, public and

private key management, access control, and certificates.

6. Software components: Known as JavaBeans, can plug into existing component architectures.

7. Object serialization: Allows lightweight persistence and communication via RMI (Remote Method Invocation).

2.5 Tools Used :

2.5.1 JDK

The Java Development Kit is an Oracle Corporation product aimed at Java developers. Since the introduction of Java, it has been by far the most widely used Java Software Development Kit (SDK). On 17 November 2006, Sun announced that it would be released under the GNU General Public License (GPL), thus making it free software. This happened in large part on 8 May 2007; Sun contributed the source code to the Open JDK.

The JDK has as its primary components a collection of programming tools, including:

1. java – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
2. javac – the compiler, which converts source code into Java bytecode
3. appletviewer – this tool can be used to run and debug Java applets without a web browser
4. apt – the annotation-processing tool.
5. extcheck – a utility which can detect JAR-file conflicts
6. idlj – the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.

7. javadoc – the documentation generator, which automatically generates documentation from source code comments
8. jar – the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
9. javah – the C header and stub generator, used to write native methods
10. javap – the class file disassembler
11. javaws – the Java Web Start launcher for JNLP applications
12. jconsole – Java Monitoring and Management Console
13. jdb – the debugger
14. jhat – Java Heap Analysis Tool
15. jinfo – This utility gets configuration information from a running Java process or crash dump.
16. jmap – This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump.
17. jps – Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system.
18. jrunscript – Java command-line script shell.
19. jstack – utility which prints Java stack traces of Java threads
20. jstat – Java Virtual Machine statistics monitoring tool
21. jstatd – jstat daemon
22. policytool – the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources
23. VisualVM – visual tool integrating several command-line JDK tools and lightweight performance and memory profiling capabilities
24. wsimport – generates portable JAX-WS artifacts for invoking a web service.
25. xjc – Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

The JDK also comes with a complete Java Runtime Environment, usually called a private runtime, due to the fact that it is separated from the "regular" JRE and has extra

contents. It consists of a Java Virtual Machine and all of the class libraries present in the production environment, as well as additional libraries only useful to developers, such as the internationalization libraries and the IDL libraries.

2.5.2 Net Beans IDE 6.9.1

Net Beans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Groovy, C, C++, Scala, Clojure, and others. The Net Beans IDE 7.0 no longer supports Ruby and Ruby on Rails, but a third party has begun work on a separate plug-in. The Net Beans IDE is written in Java and can run on Windows, Mac OS, Linux, Solaris and other platforms supporting a compatible JVM. A pre-existing JVM or a JDK is not required.

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

1. Net Beans Platform

The Net Beans Platform is a reusable framework for simplifying the development of Java Swing desktop applications. The Net Beans IDE bundle for Java SE contains what is needed to start developing Net Beans plugins and Net Beans Platform based applications; no additional SDK is required. Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally-signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

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

1. User interface management (e.g. menus and toolbars)
2. User settings management
3. Storage management (saving and loading any kind of data)
4. Window management
5. Wizard framework (supports step-by-step dialogs)
6. Net Beans Visual Library
7. Integrated development tools

Net Beans IDE is a free, open-source, cross-platform IDE with built-in-support for Java Programming Language.

2. Net Beans IDE

NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, Maven support, refactorings,version control (supporting CVS, Subversion, Mercurial and Clearcase).

All the functions of the IDE are provided by modules. Each module provides a well defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended. New features, such as support for other programming languages, can be added by installing additional modules. For instance, Sun Studio, Sun Java Studio Enterprise, and Sun Java Studio Creator from Sun Microsystems are all based on the NetBeans IDE.

From July 2006 through 2007, NetBeans IDE was licensed under Sun's Common Development and Distribution License (CDDL), a license based on the Mozilla Public License (MPL). In October 2007, Sun announced that NetBeans would

henceforth be offered under a dual license of the CDDL and the GPL version 2 licenses, with the GPL linking exception for GNU Classpath

Chapter Three

Design

3.1 Introduction

The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operational characteristics of the system.

3.2 Physical Design:

The interface of the application is very aesthetic, user friendly that even a novice user can understand and use it quickly to achieve the required task. The user has to simply follow the syntax given on the form which is very simple.

He/she simply performs various operation like mouse Move, left click or right click.

The context level DFD is here

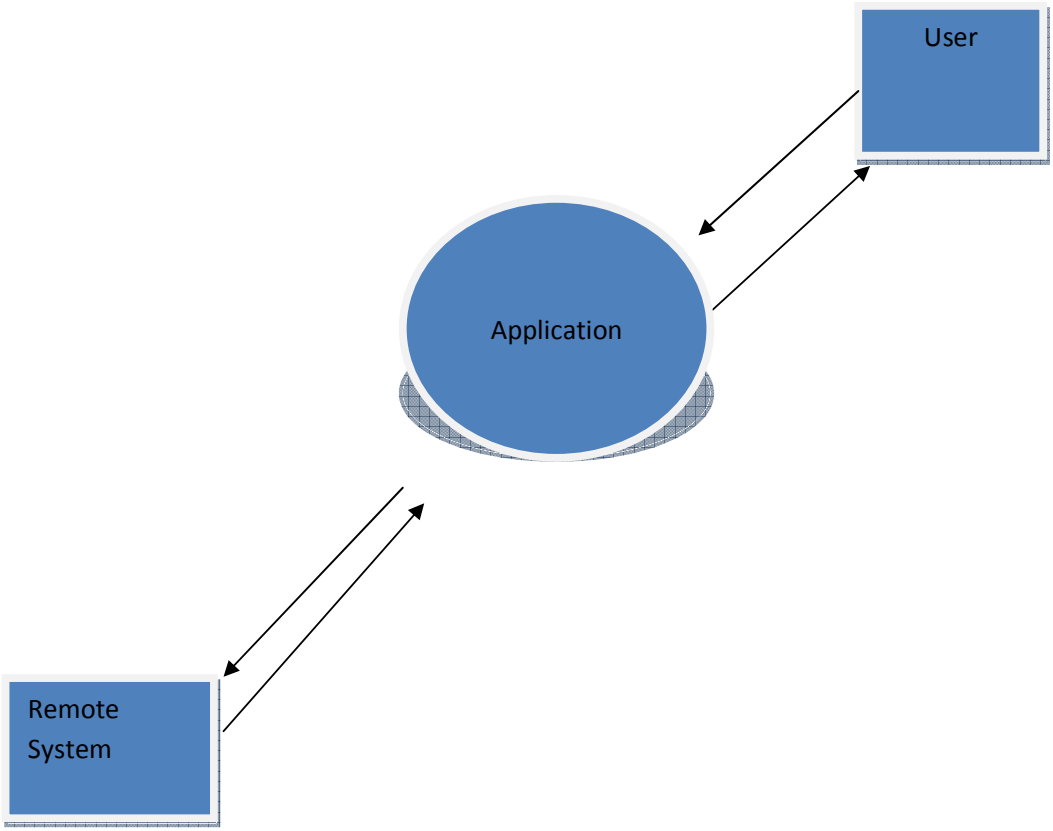


Figure 3.1: Context level DFD

Processing Logic:

The user starts the application first.

After that the user enters the ip address of the target computer.

Then the user inserts the password in the appropriate text field.

After that, the user starts the connection by clicking on the start button.

The user can see the target machine desktop on his screen.

The user can operate on it as if it is his desktop only. He can use the mouse clicks, keystrokes.

The flow of program is explained with this diagram.

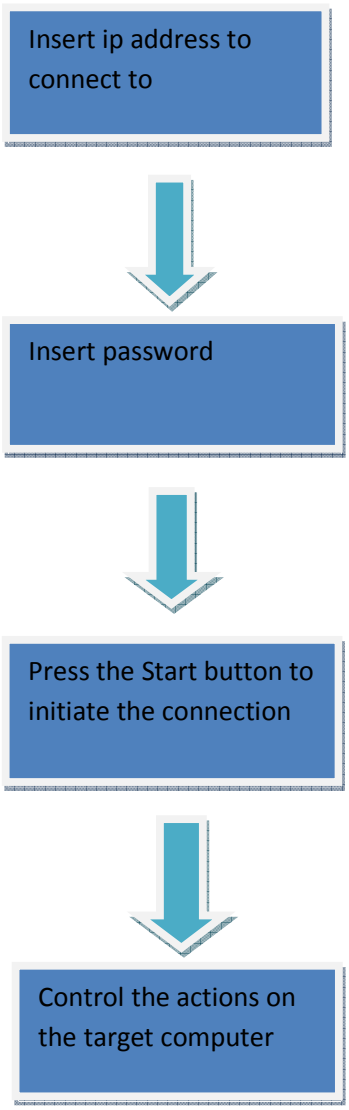


Figure 3.2: Application Processing Logic

3.3 Use Case Diagram:

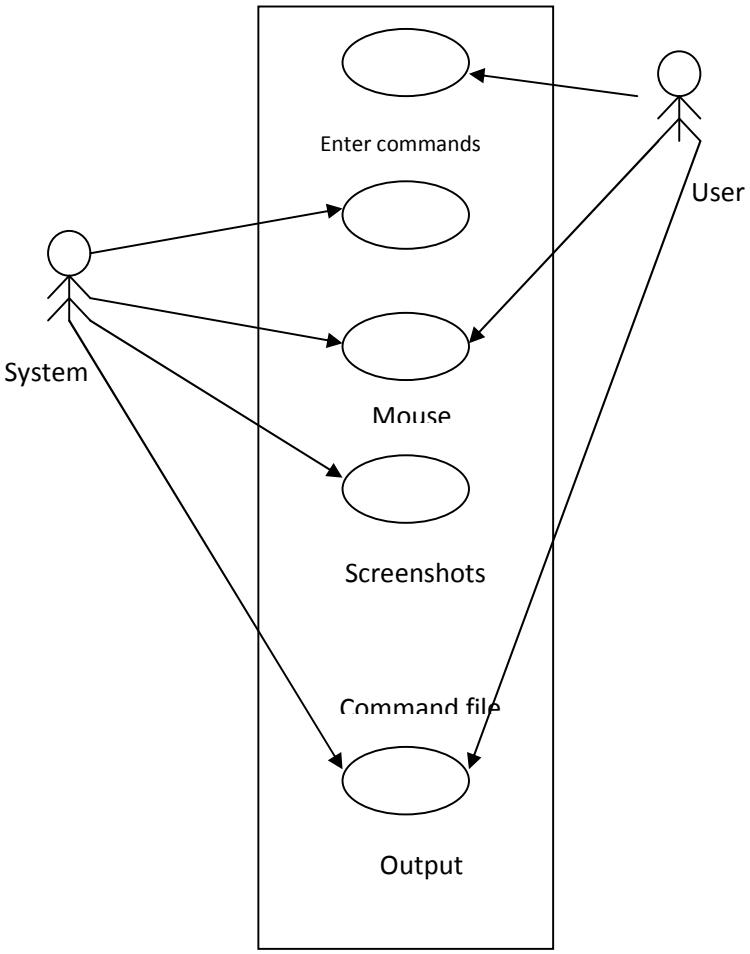


Figure 3.3: Use Case Diagram

3.4 State Chart Diagram:

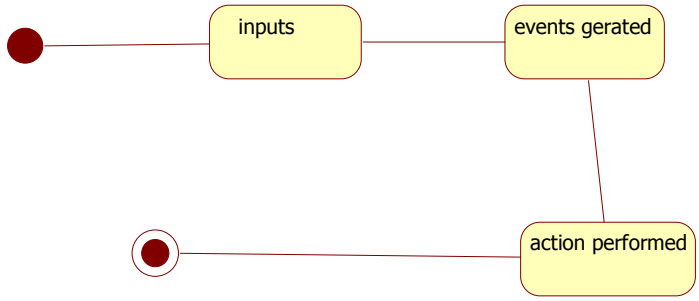


Figure 3.6: State Chart Diagram

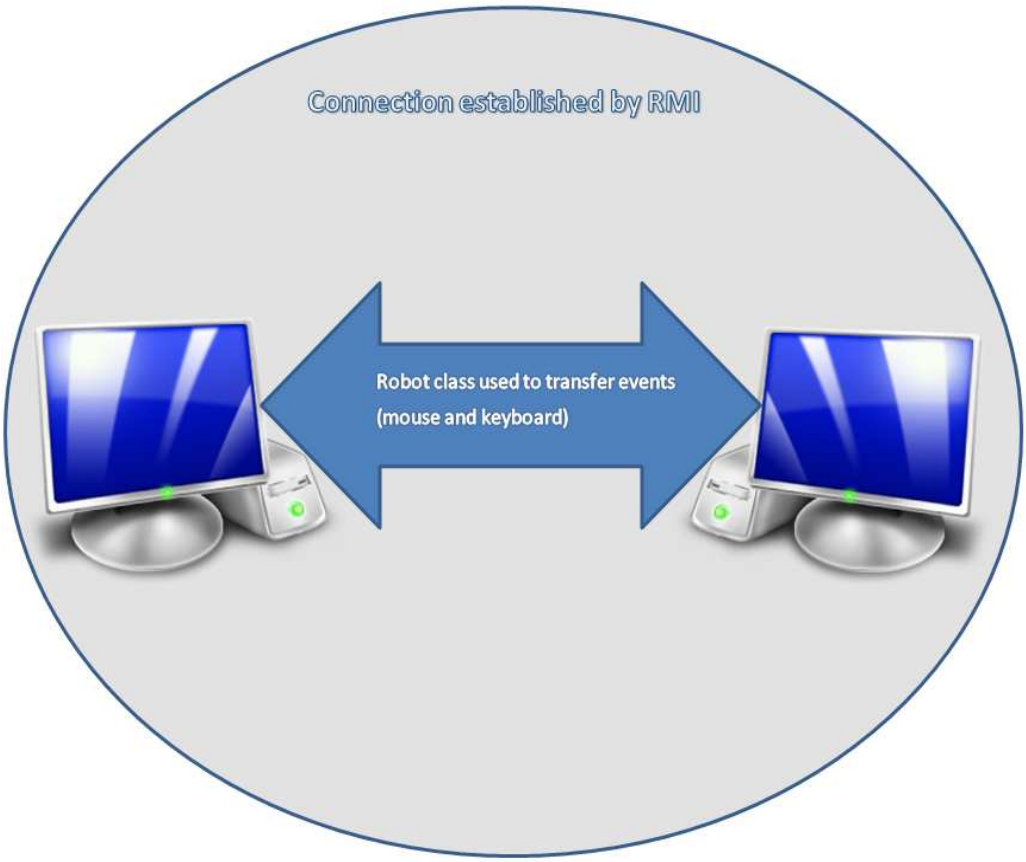


Figure3.5: Basic Idea of Project

Our Project “Remote Desktop Shaing” is based on two concepts :-

- RMI (Remote method Invocation)
- Robot Class

Here the connection establishment between client and server is done through RMI, once the connection is established, the work of the robot class starts.

Robot class helps the program to send and receive different events like

Mousevents – Mouse Button Pressed/Release, Pointer location, Scroll Wheel, etc.

Key events – Key Strokes, Key Pressed/Release.

Buffered Image – Screen capture, Pixel informaton.

Basic Idea of RMI:

Our Project is based mainly upon the remote desktop sharing.

3.6 RMI (Remote Desktop Sharing): Remote Method Invocation (RMI) facilitates object function calls between Java Virtual Machines (JVMs). JVMs can be located on separate computers - yet one JVM can invoke methods belonging to an object stored in another JVM. Methods can even pass objects that a foreign virtual machine has never encountered before, allowing dynamic loading of new classes as required.

Java RMI allows:

- provide user with a “thin client
    - allows good performance on lower end workstations
- run server on high end hardware
    - maximize investment over many clients
    - server remote from client
- Distributed network object

The general idea of RMI is:

- Instantiate an object on another machine
- Invoke methods on the remote object

3.6.1 Architecture of RMI

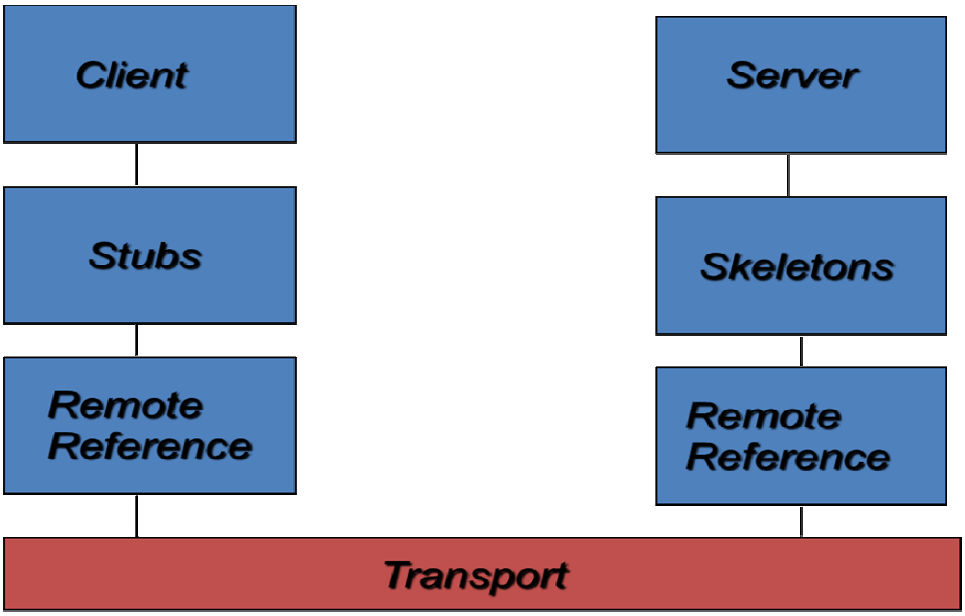


Figure 3.6: RMI Architecture

Client - user interface

Server - data source

The Stub/Skeleton Layer

The stub/skeleton layer is the interface between the application layer and the rest of the RMI system. This layer does not deal with specifics of any transport, but transmits data to the remote reference layer via the abstraction of *marshal streams*. Marshal streams employ a mechanism called *object serialization* which enables Java objects to be transmitted between address spaces. Objects transmitted using the object serialization system are passed by copy to the remote address space, unless they are remote objects, in which case they are passed by reference.

A *stub* for a remote object is the client-side proxy for the remote object. Such a stub implements all the interfaces that are supported by the remote object implementation. A client-side stub is responsible for:

- Initiating a call to the remote object (by calling the remote reference layer).
- Marshaling arguments to a marshal stream (obtained from the remote reference layer).
- Informing the remote reference layer that the call should be invoked.
- Unmarshaling the return value or exception from a marshal stream.
- Informing the remote reference layer that the call is complete.

A *skeleton* for a remote object is a server-side entity that contains a method which dispatches calls to the actual remote object implementation. The skeleton is responsible for:

- Unmarshaling arguments from the marshal stream.
- Making the up-call to the actual remote object implementation.
- Marshaling the return value of the call or an exception (if one occurred) onto the marshal stream.

The appropriate stub and skeleton classes are determined at run time and are dynamically loaded as needed. Dynamic Stub Loading describes in detail how the stubs are located and how their actions are constrained.

3.6.2 The Remote Reference Layer

The remote reference layer deals with the lower level transport interface. This layer is also responsible for carrying out a specific remote reference protocol which is independent of the client stubs and server skeletons.

Each remote object implementation chooses its own remote reference subclass that operates on its behalf. Various invocation protocols can be carried out at this layer, for example:

- Unicast point-to-point invocation.

- Invocation to replicated object groups.
- Support for a specific replication strategy.
- Support for a persistent reference to the remote object (enabling activation of the remote object).
- Reconnection strategies (if remote object becomes inaccessible).

The remote reference layer has two cooperating components: the client-side and the server-side components. The client-side component contains information specific to the remote server (or servers, if the remote reference is to a replicated object) and communicates via the transport to the server-side component. During each method invocation, the client and server-side components perform the specific remote reference semantics. For example, if a remote object is part of a replicated object, the client-side component can forward the invocation to each replica rather than just a single remote object.

In a corresponding manner, the server-side component implements the specific remote reference semantics prior to delivering a remote method invocation to the skeleton. This component, for example, could handle ensuring atomic multiple delivery by communicating with other servers in the replica group.

The remote reference layer transmits data to the transport layer via the abstraction of a stream-oriented *connection*. The transport takes care of the implementation details of connections. Although connections present a streams-based interface, a connectionless transport may be implemented beneath the abstraction.

3.6.3 The Transport Layer

In general, the transport layer of the RMI system is responsible for:

- Setting up connections to remote address spaces.
- Managing connections.

- Monitoring connection "liveness."
- Listening for incoming calls.
- Maintaining a table of remote objects that reside in the address space.
- Setting up a connection for an incoming call.
- Locating the dispatcher for the target of the remote call and passing the connection to this dispatcher.

The concrete representation of a remote object reference consists of an endpoint and an object identifier. This representation is called a *live reference*. Given a live reference for a remote object, a transport can use the endpoint to set up a connection to the address space in which the remote object resides. On the server side, the transport uses the object identifier to look up the target of the remote call.

The transport for the RMI system consists of four basic abstractions:

- An *endpoint* is the abstraction used to denote an address space or Java virtual machine. In the implementation, an endpoint can be mapped to its transport. That is, given an endpoint, a specific transport instance can be obtained.
- A *channel* is the abstraction for a conduit between two address spaces. As such, it is responsible for managing connections between the local address space and the remote address space for which it is a channel.
- A *connection* is the abstraction for transferring data (performing input/output).
- The *transport* abstraction manages channels. Each channel is a virtual connection between two address spaces. Within a transport, only one channel exists per pair of address spaces, the local address space and a remote address space. Given an endpoint to a remote address space, a transport sets up a channel to that address space. The transport abstraction is also responsible for accepting calls on incoming connections to the address space, setting up a connection object for the call, and dispatching to higher layers in the system.

A transport defines what the concrete representation of an endpoint is, so multiple transport implementations may exist. The design and implementation also supports multiple transports per address space, so both TCP and UDP can be supported in the same virtual machine.

The steps involved in the RMI process are:

- Create the Interface to the server
- Create the Server
- Create the Client
- Compile the Interface (javac)
- Compile the Server (javac)
- Compile the Client (javac)
- Generate Stubs and Skeletons (rmic)

RMI Registry-The RMI Registry is a naming service provided with the JDK as a teaching tool or for a small number of Remote Objects

- Uses port 1099 as its default port
- Can be considered to be a reference implementation
- runs out of steam above a 100 objects
- runs on same machine as the remote object
- Use another naming service
- J2EE uses JNDI and Directory Services to provide a more robust naming service

- Silver stream uses JNDI with its own Service Provider and repository for it

3.6.4 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. For example, X-Window systems will throw the exception if the XTEST 2.2 standard extension is not supported (or not enabled) by the X server.

Some methods and Constructors

Constructor Summary
Robot() Constructs a Robot object in the coordinate system of the primary screen.
Robot(GraphicsDevice screen) Creates a Robot for the given screen device.

Method Summary

BufferedImage	**createScreenCapture**(Rectangle screenRect) Creates an image containing pixels read from the screen.
void	**delay**(int ms) Sleeps for the specified time.
int	**getAutoDelay**() Returns the number of milliseconds this Robot sleeps after generating an event.
Color	**getPixelColor**(int x, int y) Returns the color of a pixel at the given screen coordinates.
boolean	**isAutoWaitForIdle**() Returns whether this Robot automatically invokes `waitForIdle` after generating an event.
void	**keyPress**(int keycode) Presses a given key.
void	**keyRelease**(int keycode) Releases a given key.
void	**mouseMove**(int x, int y) Moves mouse pointer to given screen coordinates.
void	**mousePress**(int buttons) Presses one or more mouse buttons.
void	**mouseRelease**(int buttons) Releases one or more mouse buttons.
void	**setAutoDelay**(int ms) Sets the number of milliseconds this Robot sleeps after generating an event.
void	**setAutoWaitForIdle**(boolean isOn) Sets whether this Robot automatically invokes `waitForIdle` after generating an event.
String	**toString**() Returns a string representation of this Robot.
void	**waitForIdle**() Waits until all events currently on the event queue have been processed.

Chapter Four

FEASIBILITY ANALYSIS

4.1 Aim

The main aim of feasibility study is to determine whether development of the application is financially and technically feasible or not. It involves the analysis of problem and collection of data which will be put into the system, the processing required to be carried out on this data, the output required to be produced by the system as well as the study of various constraints on the behaviour of the system.

The application can be developed in many languages like Visual C++, Visual Basic, and Java etc. But Java is the most suitable language for the development of these applications it is platform independent, takes less time in execution as far as network application is concerned. It also contains various packages that contain various classes that are needed to implement the functions for the application. The packages that are to be included in this application are Java. Swing, Java.awt, Java.awt.event. Hence, Java provides all the methods that are to be used in this application. Moreover much help regarding Java is available on the internet, so the problem regarding the implementation will be very less.

The application to be developed does not require any high investment. The software required JDK 1.4 is easily available in the market. Hence financially also the project is very much feasible.

Hence, from the above stated points, the application seams to be very much feasible, both technically and financially.

An Important outcome of the preliminary investigation is the determination that the system requested is feasible. There are 3 aspects in the feasibility study:

4.2 Technical Feasibility

It is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may vary considerably, but might

include:

The facility to produce outputs in a given time.

1) Response time under certain conditions.
3) Ability to process a certain volume of transaction at a articular speed.
4) Facility to communicate data to distant location.

In examining technical feasibility, configuration of the system is given more importance than the actual make of hardware. The configuration should given the complete picture about the system's requirements like how many workstations are required, how these units are interconnected so that they could operate and communicate smoothly. What speeds of input and output should be achieved at particular quality of printing. This can be used b e used as a basis for the tender document against which dealers and manufactures can later make their equipment bids. Specific hardware and software products can then be evaluated keeping in view with the logical needs.

At the feasibility stage it is desirable that two or three different configurations will be pursued that satisfy the key technical requirements but which represent different levels of ambitions and cost. Investigation of these technical alternatives can be aided by approaching a range of sup pliers for preliminary discussions. Out of all types of feasibility, technical feasibility generally is the most difficult to determine.

4.3 Operational Feasibility

It is mainly related to human organizational and political aspects. The points to be considered are:

1. What changes will be brought with the system?
2. What organizational structures are distributed?
3. What new skills will be required? Do existing staff members have these skills? If not, can they be trained in due course of time?

Generally project will not be rejected simply because of operational infeasibility but such considerations are likely to critically affect the nature and scope of the eventual recommendations. This feasibility study is carries out by a small group of people who are familiar with information system techniques, who understand the parts of the business that are relevant to the project and are skilled in system analysis and design process.

As far as this project of Personal Information System is concerned the changes which we have to be brought were only organizational. Then our focus goes towards workstations. Keeping in view of their hardware requirements like network interface card etc.

4.4 Economical Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost/benefit analysis; the procedure is to determine the benefits and savings that expected from a proposed system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an on going effort that improves in accuracy at each phase of the system life cycle.

This feasibility also depends upon quality of staff hired and the proposed duration of time taken in this project sometimes it might be possible due to extension of time duration may fall the project under loss. The study of feasibility changes from phase to phase of the project development.

In this project although this feasibility study doesn't matter much in the case new setup of project but on the other hand if we have to modify over existing system we must take care of our existing resources and must analyse specially the working condition of hardware.

The main components of making software are:

1. System and software requirements analysis
2. Design and implementation software
3. Ensuring, verifying and maintaining software integrity. System Analysis is an activity that encompasses most of the tasks that are called Computer System Engg. Confusion sometimes occurs because the term is often used in context that alludes it only to software requirement analysis activities, but system analysis focuses on all the system elements -not just software system analysis is conducted with the following objectives in mind.
4. Identify the Need
5. Evaluate the system concept for feasibility
6. Perform economic and technical analysis
7. Allocate functions to hardware, software, database and other system elements
8. Establish cost and schedule constraints
9. Create a system definition that forms the foundation for all the subsequent e.g. Word

Lines Of Code	550
Duration	4-5 Months

Chapter Five

Implementation

Server Code

1. ClientCommandsSender.java

```
package serverdesktop;
import java.awt.Rectangle;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import javax.swing.JPanel;

class ClientCommandsSender implements KeyListener,
      MouseMotionListener,MouseListener {

private Socket cSocket = null;
private JPanel cPanel = null;
private PrintWriter writer = null;
private Rectangle clientScreenDim = null;

ClientCommandsSender(Socket s, JPanel p, Rectangle r) {
cSocket = s;
cPanel = p;
clientScreenDim = r;
```

```
cPanel.addKeyListener(this);
cPanel.addMouseListener(this);
cPanel.addMouseMotionListener(this);
try {
writer = new PrintWriter(cSocket.getOutputStream());
        } catch (IOException ex) {
ex.printStackTrace();
        }
    }
public void mouseDragged(MouseEvent e) {
    }
public void mouseMoved(MouseEvent e) {
double xScale = clientScreenDim.getWidth()/cPanel.getWidth();
System.out.println("xScale: " + xScale);
double yScale = clientScreenDim.getHeight()/cPanel.getHeight();
System.out.println("yScale: " + yScale);
System.out.println("Mouse Moved");
writer.println(EnumCommands.MOVE_MOUSE.getAbbrev());
writer.println((int)(e.getX() * xScale));
writer.println((int)(e.getY() * yScale));
writer.flush();
    }
public void mouseClicked(MouseEvent e) {
    }
public void mousePressed(MouseEvent e) {
System.out.println("Mouse Pressed");
writer.println(EnumCommands.PRESS_MOUSE.getAbbrev());
```

```
int button = e.getButton();
int xButton = 16;
if (button == 3) {
xButton = 4;
        }
writer.println(xButton);
writer.flush();
    }
public void mouseReleased(MouseEvent e) {
System.out.println("Mouse Released");
writer.println(EnumCommands.RELEASE_MOUSE.getAbbrev());
int button = e.getButton();
int xButton = 16;
if (button == 3) {
xButton = 4;
        }
writer.println(xButton);
writer.flush();
    }
public void mouseEntered(MouseEvent e) {
    }
public void mouseExited(MouseEvent e) {
    }
public void keyTyped(KeyEvent e) {
    }
public void keyPressed(KeyEvent e) {
System.out.println("Key Pressed");
```

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

2. Clienthandler.java

```
package serverdesktop;
import java.awt.BorderLayout;
import java.awt.Rectangle;
import java.beans.PropertyVetoException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.Socket;
import javax.swing.JDesktopPane;
import javax.swing.JInternalFrame;
import javax.swing.JPanel;

class ClientHandler extends Thread {
```

```
private JDesktopPane desktop = null;
private Socket cSocket = null;
private JInternalFrame interFrame = new JInternalFrame("Client Screen",
true, true, true);
private JPanel cPanel = new JPanel();
public ClientHandler(Socket cSocket, JDesktopPane desktop) {
        this.cSocket = cSocket;
        this.desktop = desktop;
start();
    }
public void drawGUI(){
interFrame.setLayout(new BorderLayout());
interFrame.getContentPane().add(cPanel,BorderLayout.CENTER);
interFrame.setSize(100,100);
desktop.add(interFrame);
try {
interFrame.setMaximum(true);
        } catch (PropertyVetoException ex) {
ex.printStackTrace();
        }
cPanel.setFocusable(true);
interFrame.setVisible(true);
    }
public void run(){
        Rectangle clientScreenDim = null;
```

```
        ObjectInputStream ois = null;
drawGUI();
try{
ois = new ObjectInputStream(cSocket.getInputStream());
clientScreenDim =(Rectangle) ois.readObject();
}catch(IOException ex){
ex.printStackTrace();
}catch(ClassNotFoundException ex){
ex.printStackTrace();
        }
new ClientScreenReciever(ois,cPanel);
new ClientCommandsSender(cSocket,cPanel,clientScreenDim);
    }
}
```

3. ClientScreenReciever.java

```
package serverdesktop;
import java.awt.Graphics;
import java.awt.Image;
import java.io.IOException;
import java.io.ObjectInputStream;
import javax.swing.ImageIcon;
import javax.swing.JPanel;
class ClientScreenReciever extends Thread {
```

```
private ObjectInputStream cObjectInputStream = null;
private JPanel cPanel = null;
private boolean continueLoop = true;
public ClientScreenReciever(ObjectInputStream ois, JPanel p) {
cObjectInputStream = ois;
cPanel = p;
start();
   }
public void run(){
try {
while(continueLoop){
                ImageIcon imageIcon = (ImageIcon) cObjectInputStream.readObject();
System.out.println("New image recieved");
                Image image = imageIcon.getImage();
image = image.getScaledInstance(cPanel.getWidth(),cPanel.getHeight()
,Image.SCALE_FAST);
                Graphics graphics = cPanel.getGraphics();
graphics.drawImage(image, 0, 0, cPanel.getWidth(),cPanel.getHeight(),cPanel);
              }
          } catch (IOException ex) {
ex.printStackTrace();
        } catch(ClassNotFoundException ex){
ex.printStackTrace();
        }
    }
```

```
}
```

4.EnumCommands.java

```
package serverdesktop;
public enum EnumCommands {
   PRESS_MOUSE(-1),
   RELEASE_MOUSE(-2),
   PRESS_KEY(-3),
   RELEASE_KEY(-4),
   MOVE_MOUSE(-5);
private int abbrev;
EnumCommands(int abbrev){
       this.abbrev = abbrev;
   }
public int getAbbrev(){
return abbrev;
   }
}
```

5.Main.java

```
package serverdesktop;
public class Main {
```

```
public static void main(String[] args) {
    }
}
```

6.ServerInitiator.java

```
package serverdesktop;
import java.awt.BorderLayout;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import javax.swing.JDesktopPane;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class ServerInitiator {
private JFrame frame = new JFrame();
private JDesktopPane desktop = new JDesktopPane();

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

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

```
drawGUI();
while (true) {
            Socket client = sc.accept();
System.out.println("New client Connected to the server");
new ClientHandler(client, desktop);
          }
      } catch (IOException ex) {
ex.printStackTrace();
      }
   }
public void drawGUI() {
frame.add(desktop, BorderLayout.CENTER);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      //Show the frame in a maximized state
frame.setExtendedState(frame.getExtendedState() | JFrame.MAXIMIZED_BOTH);
frame.setVisible(true);
   }
}
```

7.Server.java

```
package serverdesktop;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
```

```
import java.io.*;


public class server extends Frame implements ActionListener, Runnable
{
      Image Icon = Toolkit.getDefaultToolkit().getImage("hi.gif");
      ServerSocket ss;
      Socket s;
         BufferedReader br;
         BufferedWriter bw;
         TextField text;
      Button sendBut, exitBut;
      List list;
public server(String m) // class constructor
         {
super(m);
setSize(300, 130);
setLocation(0,0);
setIconImage(Icon);
setResizable(false);
setBackground(new Color(192, 192, 192));
this.setLayout(new GridLayout(2, 1));
            Panel panels[] = new Panel[2];
panels[0] = new Panel();
panels[1] = new Panel();
panels[0].setLayout(new BorderLayout());
```

```
panels[1].setLayout(new FlowLayout(FlowLayout.LEFT));
sendBut = new Button("Send");
exitBut = new Button("Exit");
sendBut.addActionListener(this);
exitBut.addActionListener(this);
list = new List();
list.addItem("Server up & Listening on port plz wait...");
text = new TextField(25);
panels[0].add(list);
panels[1].add(text);
panels[1].add(sendBut);
panels[1].add(exitBut);
add(panels[0]);
add(panels[1]);
setVisible(true);
try
        {
ss = new ServerSocket(1053);//some port number, better be above 1000
            s = ss.accept();
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
bw = new BufferedWriter(new OutputStreamWriter(s.getOutputStream()));
bw.write("Hi! ASL plz??");
bw.newLine();
bw.flush();
            Thread th;
```

```
th = new Thread(this);
th.start();
                }catch(Exception e){}
        }
server() {
throw new UnsupportedOperationException("Not yet implemented");
   }
public void run()
        {
while (true)
                {
try
                {
list.addItem(br.readLine());
}catch (Exception e){}
                }
        }
public static void main(String arg[])
        {
new server("Server Applicaton");
        }
public void actionPerformed(ActionEvent ae)
        {
if (ae.getSource().equals(exitBut))
                        System.exit(0);
```

```
else
            {
try
                {
bw.write(text.getText());
bw.newLine();bw.flush();
text.setText("");
}catch(Exception x){}
                 }
        }
}
```

Client Source

1.Client initiator.java

```
package cleintdesktop;
import java.awt.AWTException;
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.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class ClientInitiator {
public void initialize() {
    Socket socket = null;
    String ip = JOptionPane.showInputDialog("Please enter server IP");
```

```
        String port_no = JOptionPane.showInputDialog("Please enter server port");
int port = Integer.parseInt(port_no);
        Robot robot = null; //Used to capture the screen
        Rectangle rectangle = null; //Used to represent screen dimensions
try {
System.out.println("Connecting to server .....");
socket = new Socket(ip, port);
System.out.println("Connection Established.");
      GraphicsEnvironment gEnv=
GraphicsEnvironment.getLocalGraphicsEnvironment();
           GraphicsDevice gDev = gEnv.getDefaultScreenDevice();
          Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
rectangle = new Rectangle(dim);
robot = new Robot(gDev);
drawGUI();
new ScreenSpyer(socket, robot, rectangle);
new ServerDelegate(socket, robot);
        } catch (UnknownHostException ex) {
ex.printStackTrace();
        } catch (IOException ex) {
ex.printStackTrace();
        } catch (AWTException ex) {
ex.printStackTrace();
        }
    }
```

```
private void drawGUI() {
        JFrame frame = new JFrame("Remote Admin");
        JButton button = new JButton("Terminate");
frame.setBounds(100, 100, 150, 150);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.add(button);
button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
System.exit(0);
            }
        });
frame.setVisible(true);
    }
}
```

2.EnumCommands.java

```
package cleintdesktop;
public enum EnumCommands {
    PRESS_MOUSE(-1),
    RELEASE_MOUSE(-2),
    PRESS_KEY(-3),
    RELEASE_KEY(-4),
    MOVE_MOUSE(-5);
private int abbrev;
```

```
EnumCommands(int abbrev){
        this.abbrev = abbrev;
    }
public int getAbbrev(){
return abbrev;
    }
}
```

3.Main.java

```
package cleintdesktop;
public class Main {
public static void main(String[] args) {
    }
}
```

4.Screenspyer.java

```
package cleintdesktop;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.Socket;
import javax.swing.ImageIcon;
```

```
class ScreenSpyer extends Thread {
   Socket socket = null;
   Robot robot = null; // Used to capture screen
   Rectangle rectangle = null; //Used to represent screen dimensions
boolean continueLoop = true; //Used to exit the program
public ScreenSpyer(Socket socket, Robot robot,Rectangle rect) {
       this.socket = socket;
       this.robot = robot;
rectangle = rect;
start();
   }
public void run(){
       ObjectOutputStream oos = null; //Used to write an object to the streem
try{
oos = new ObjectOutputStream(socket.getOutputStream());
oos.writeObject(rectangle);
}catch(IOException ex){
ex.printStackTrace();
       }
while(continueLoop){
           BufferedImage image = robot.createScreenCapture(rectangle);
           ImageIcon imageIcon = new ImageIcon(image);
try {
System.out.println("before sending image");
oos.writeObject(imageIcon);
```

```
oos.reset(); //Clear ObjectOutputStream cache
System.out.println("New screenshot sent");
            } catch (IOException ex) {
ex.printStackTrace();
            }
try{
Thread.sleep(100);
}catch(InterruptedException e){
e.printStackTrace();
            }
        }
    }
}
```

5.Serverdelegate.java

```
package cleintdesktop;
import java.awt.Robot;
import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;
class ServerDelegate extends Thread {
    Socket socket = null;
    Robot robot = null;
boolean continueLoop = true;
```

```
public ServerDelegate(Socket socket, Robot robot) {
        this.socket = socket;
        this.robot = robot;
start(); //Start the thread and hence calling run method
    }
public void run(){
        Scanner scanner = null;
try {
System.out.println("Preparing InputStream");
scanner = new Scanner(socket.getInputStream());
while(continueLoop){
System.out.println("Waiting for command");
int command = scanner.nextInt();
System.out.println("New command: " + command);
switch(command){
case -1:
robot.mousePress(scanner.nextInt());
break;
case -2:
robot.mouseRelease(scanner.nextInt());
break;
case -3:
robot.keyPress(scanner.nextInt());
break;
case -4:
```

```
robot.keyRelease(scanner.nextInt());
break;
case -5:
robot.mouseMove(scanner.nextInt(), scanner.nextInt());
break;
            }
          }
      } catch (IOException ex) {
ex.printStackTrace();
      }
   }
```

6. Client.java

```
package cleintdesktop;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
public class client extends Frame implements ActionListener, Runnable
{
      Image Icon = Toolkit.getDefaultToolkit().getImage("hi.gif") ;
         Socket s;
         BufferedReader br;
         BufferedWriter bw;
```

```
        TextField text;
      Button sendBut, exitBut;
        List list;
public client(String st)
        {
super(st);
setSize(300, 130);
                setIconImage(Icon);
                setLocation(300,0);
setResizable(false);
setBackground(new Color(192, 192, 192));
                this.setLayout(new GridLayout(2, 1));
            Panel panels[] = new Panel[2];
panels[0] = new Panel();
panels[1] = new Panel();
panels[0].setLayout(new BorderLayout());
panels[1].setLayout(new FlowLayout(FlowLayout.LEFT));
sendBut = new Button("Send");
exitBut = new Button("Exit");
sendBut.addActionListener(this);
exitBut.addActionListener(this);
list = new List();
                text = new TextField(25);
panels[0].add(list);
panels[1].add(text);
```

```
panels[1].add(sendBut);
panels[1].add(exitBut);
add(panels[0]);
add(panels[1]);
                    setVisible(true);
try
              {
                     s = new Socket("192.168.0.2", 1053);
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
bw = new BufferedWriter(new OutputStreamWriter(s.getOutputStream()));
                              Thread th;
                              th = new Thread(this);
                              th.start();
                    }catch(Exception e){}
          }
public static void main(String arg[])
          {
new client("Client Application");
          }
private client() {
throw new UnsupportedOperationException("Not yet implemented");
   }
public void run()
          {
while (true)
```

```
                {
                        try
                {
list.addItem(br.readLine());
                        }catch (Exception h){}
                }
        }
public void actionPerformed(ActionEvent ae)
        {
if(ae.getSource().equals(exitBut))
                        System.exit(0);
                else
             {
try
                {
bw.write(text.getText());
bw.newLine();
bw.flush();
text.setText("");
}catch(Exception m){}
                 }

         }
        {
java.awt.EventQueue.invokeLater(new Runnable() {
```

```
public void run() {
new client().setVisible(true);
            }
        });
    }
}
```

Chapter Six

Testing

6.1 Testing Overview

Testing plays a critical role in quality assurance for software .Due to the limitation of the verification method for the previous phases, design and requirement fault also appear in the code. Testing is used to detect these errors, in addition to the error introduced during coding phase.

Testing is a dynamic method for verification and validation, where the system is to be tested is executed and behaviour of the system is observed. Due to this testing the failure of the system can be observed, from which the presence of fault can be deduced. However, separate activities have to be performed to identify the faults.

There are two method of testing: functional and structural. In functional testing, the internal logic of the system under testing is not considered and the test cases are decided from the specification or the requirements. It is often called ―Black Box Testing‖. Equivalence class partitioning, boundary analysis, and cause effect graphing are examples of methods for selecting test cases for functional testing. In structural testing, the test cases are decided entirely on the internal logic of the program or module being tested.

As the goal of testing is to detect any errors in the programs different flavour of testing are often used. Unit testing are used to test a module or a small collection of modules and the focus is on detecting coding errors in modules. During integration testing modules are combined into sub-system, which are then tested. The goal here is to test the system design. In system testing and acceptance testing, the entire System is tested. The goal here is to test the requirement themselves. Structural testing can be used for unit testing while at higher level mostly functional testing is used.

In the project Monthly Materialization Report System we used the unit testing and functional testing. Testing can be done with test data, which attempts to simulate all possible conditions that may arise during processing. The plane for testing are prepared and then implemented.

The testing methods adopted in the testing of the system were Independent Unit Testing and System Testing

6.2 Independent Unit Test

IUT focuses first on the modules, independently of one another, to locate errors. This enables the tester to detect errors in coding and logic that are contained within that module alone. Those resulting from the interaction between modules are initially avoided.

IUT is generally white box oriented which is predicted on the close examination of procedural detail. It exercises all the logical decisions on their true and false side, executes all loops at their boundaries and within their operational bounds and checks whether the required validations have been met. White box testing exercises internal data structure to assure their validity.

6.3 System Testing

Here the system testing involved is the most widely used testing process consists of five stages as shown in the figure. In general, the sequence of testing activities is component testing, integration testing then user testing. However, as defects are discovered at any one stage, they required program modifications to correct them and this may require other stages in the testing process to be repeated.

However, as defects are discovered at any one stage, they require program modifications to correct them and this may require other stages in the testing process to be repeated. Errors in program components, say may come to light at a later stage of the testing process. The process is therefore an iterative one with information being fed back from later stages to earlier parts of the process.

The Testing Process

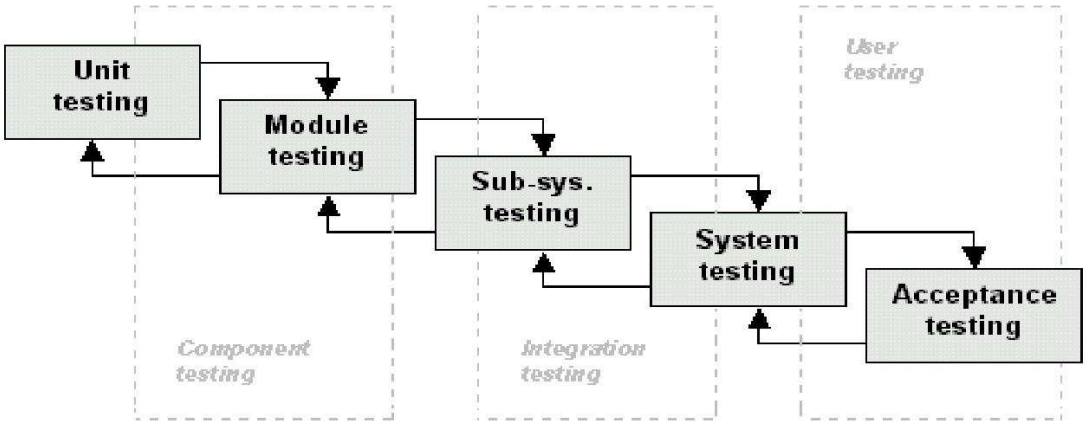


FIG 6.1 Testing process

The stages in the testing process are as follows:

6.3.1 Unit testing: (Code Oriented)

Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

6.3.2 Module testing:

A module is a collection of dependent components such as an object class, an abstract data type or some looser collection of procedures and functions. A module encapsulates related components so it can be tested without other system modules.

6.3.3 Sub-system testing: (Integration Testing) (Design Oriented)

This phase involves testing collections of modules, which have been integrated into sub-systems. Sub-systems may be independently designed and implemented. The most common problems, which arise in large software systems, are sub-systems interface mismatches. The sub-system test process should therefore concentrate on the detection

of interface errors by rigorously exercising these interfaces.

System testing:

The sub-systems are integrated to make up the entire system. The testing process is concerned with finding errors that result from unanticipated interactions between sub-systems and system components. It is also concerned with validating that the system meets its functional and non-functional requirements.

6.4 Acceptance testing:

This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system client rather than simulated test data. Acceptance testing may reveal errors and omissions in the systems requirements definition (user – oriented) because real data exercises the system in different ways from the test data. Acceptance testing may also reveal requirement problems where the system facilities do not really meet the user's needs (functional) or the system performance (non-functional) is unacceptable.

6.5 Testing Strategies:

Strategy is a general approach rather than a method of devising particular systems for component tests. Different strategies may be adopted depending on the type of system to be tested and the development process used. The testing strategies are:

6.6 Performance testing

This is used to test the run-time performance of software.

6.7Security testing

This attempt to verify that protection mechanisms built into system will protect it from improper penetration.

6.8 Recovery testing

This forces software to fail in a variety ways and verifies that recovery is properly

performed.
Large systems are usually tested using a mixture of these strategies rather than any single approach. Different strategies may be needed for different parts of the system and at different stages in the testing process.

Whatever testing strategy is adopted, it is always sensible to adopt an incremental approach to sub-system and system testing. Rather than integrate all components into a system and then start testing, the system should be tested incrementally. Each increment should be tested before the next increment is added to the system. This process should continue until all modules have been incorporated into the system.

When a module is introduced at some stage in this process, tests, which were previously unsuccessful, may now, detect defects. These defects are probably due to interactions with the new module. The source of the problem is localized to some extent, thus simplifying defect location and repair.

6.9 Equivalence Partitioning

Equivalence partitioning is a black box testing method that divides the input domain of a program into classes of data from which test cases can be derived. A typical test case uncovers a class of errors that might otherwise require many more test cases before the error is observed.

Equivalence classes for input determine the valid and invalid inputs for the program.

Equivalence class test cases are generated using the following guidelines:

If an input class specifies a range then one valid and two invalid equivalence classes are defined.

If an input class specifies a value then one valid and one invalid equivalence classes are defined.

Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once.

Table 6.1: Compatibility test OS details

Microsoft Windows 98, 2000	Owner: Mr. Jasdeep Singh Configuration: Intel P4 processor, 512MB RAM Server:
Microsoft windows 8	Owner: Mr.Manmeet Singh Monga Configuration: Intel core i5 processor, 4 GB RAM Server:
Ubuntu	Owner: Ms. Amandeep Kaur Configuration: Intel P4 processor, 512MB RAM Server:
Apple Mac OS X	Owner: Ms. Sheetal Kaur Configuration: Intel Core Duo processor, 1GB RAM Server :

Table 6.2: Compatibility test for OS

Test Subject	**Test Method**	**Expected Result**	**Actual Result**	**Remarks**
Microsoft Windows 98, 2000	Deployment of the system, black box testing	Suitable deployment, appropriate output on black box testing, average performance	System deployed and Passed black box testing with above average performance	More than expected result. Test successful.
Microsoft Vista	Deployment of the system, black box testing	Suitable deployment, appropriate output on black box testing, Best performance	System deployed and Passed black box testing with best performance	Test case successful
Ubuntu 6.04	Deployment of the system,	Suitable deployment, appropriate output on	System deployed and Passed black box	Test successful

	black box testing	black box testing, above average performance	testing with best performance	
Apple Mac OS X	Deployment of the system, black box testing	Suitable deployment, appropriate output on black box testing, above average performance	System deployed with minor tweaks that hindered performance. Passed black box testing with average performance.	The code was reconsidered and rectified. Performance increased in 2nd iteration.

Chapter Seven

Maintenance

With project management points of view it may be ranked among medium-high when we consider the whole system to be develop. The project planning, research and corresponding analysis part went efficiently. The designing of the system was middle-of-the-road that eventually lead to interruption in implementation part.

The stoppage in designing phase has leaded the system to bind the functionalities and thus reducing the scope. As the developer was short on time the extraordinary functionalities to be incorporated in the system were trimmed.

The only thing that could have been done improved would definitely be the time management. Also there has not been any similar system has been previously developed in the similar domain, therefore the developer required widespread research in similar domain of smart mail presentation developments. Due to the lack of experience in completing a project of such a scale alone, most of the time set in the Gantt chart is pure assumptions, which soon very much affects the rest of the project progress.

As the number of computer-based systems, grieve libraries of computer software began to expand. In the house of developed projects produced tones of thousand software program statements. Software products purchased from the outside added hundreds and thousands of new statements. A dark cloud appeared on the horizon. All of these programs, all of these source statements had to be corrected when false were detected, modified as user requirements changed, or adapted to new hardware that was purchased. These activities were collectively called software maintenance.

The Maintenance phase focuses on change that is associated with error correction, adaptations required as the software environment evolves, and changes due to enhancements brought by changing user requirements. Four types of changes are encountered during maintenance phase.

1. Correction
2. Adaptation

3. Enhancement
4. Prevention

1. Correction:

Even with the best quality assurance activities is lightly that the user will uncover defects in the software. Corrective maintenance changes the software to correct defects.

Maintenance is a set of Software Engineering activities that occur after software has been delivered to the user and put into operation. Software Configuration Management is a set of tracking and control activities that began when a software project begins and terminates only when the software is taken out of the operation. We may define maintenance by describing four activities that are undertaken after a program is released for use.

1. Corrective maintenance
2. Adaptive Maintenance
3. Perfective Maintenance or Enhancement
4. Preventive Maintenance or Re-engineering.

2. Adaptation

Over time, the original environment (E > G, CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance results in modification of the software to accommodate change to its external environment.

3. Enhancement

As software is used, the user will recognize additional functions that will provide benefit. Perceptive maintenance extends the software beyond the original functional requirements.

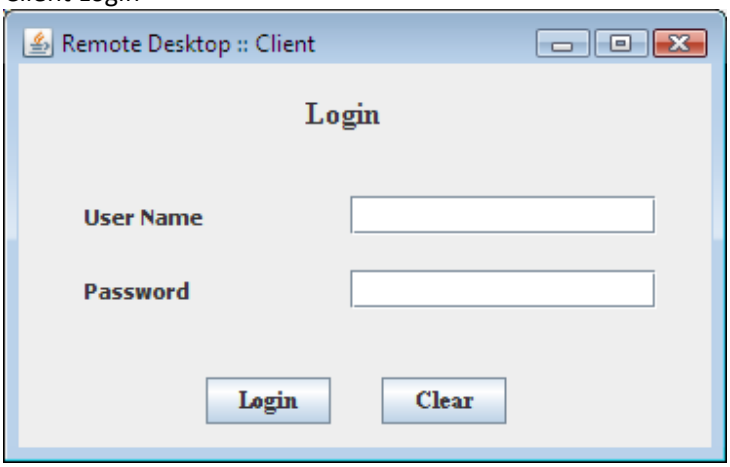
4. Prevention

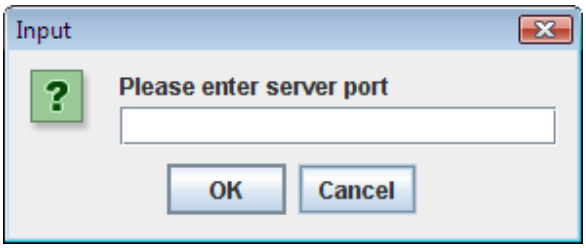
Computer Software deteriorates due to change, and because of this, preventive maintenance, often called Software Engineering, and must be conducted to enable the software to serve the needs of its end users .In essence, preventive maintenance makes changes to computer diagrams so that they can be more easily corrected, adapted, and enhanced. Software configuration Management (SCM) is an umbrella activity that is applied throughput the software process. SCM activities are developed to Identify Change .Control change. Ensure that change is being properly implemented .Report change to others that may have an interest.

APPENDIX A

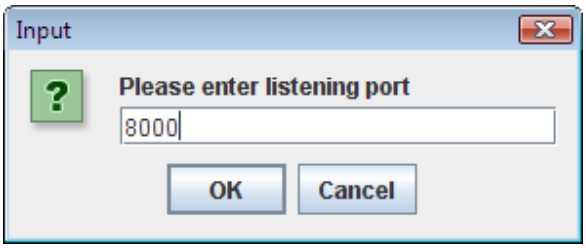
Screen Shot

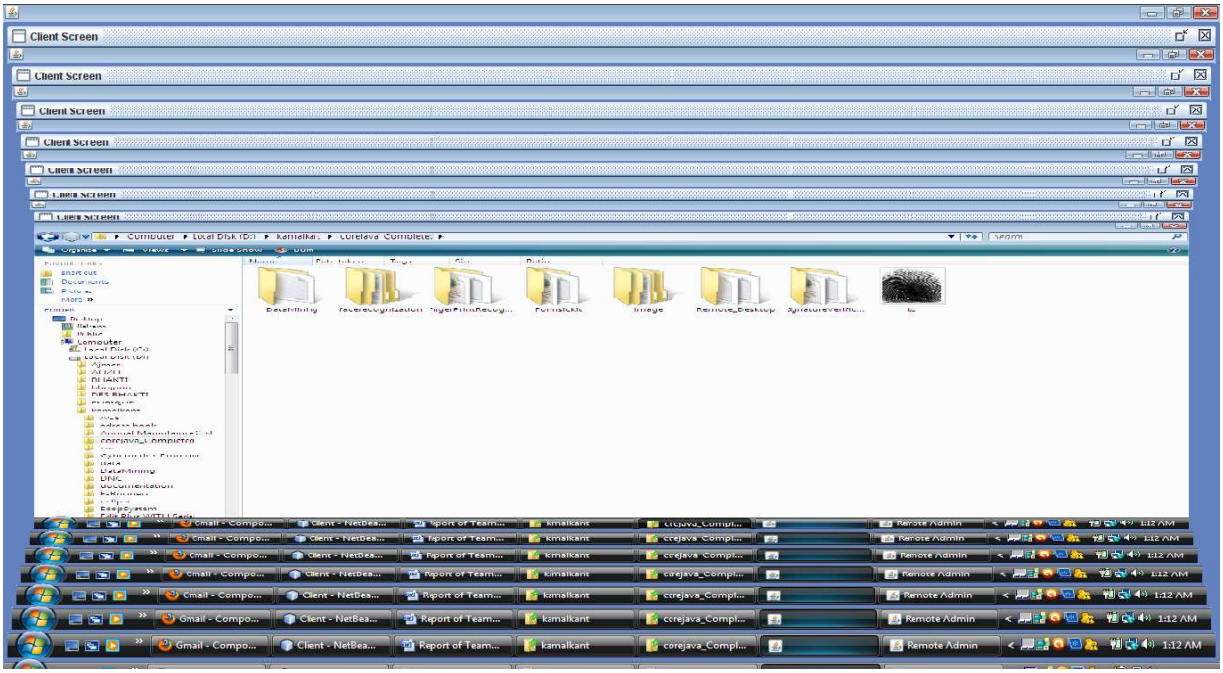
Client Login





Server side


Input
Please enter listening port
8000
OK
Cancel


Client Screen
Client Screen
Client Screen
Client Screen
Client Screen
Client Screen
Client Screen
Client Screen
Gmail - Compo...
Client - NetBea...
Report of Team...
kamalkant
corejava_Compl...
Remote Admin
1:12 AM