ONLINE MONITORING

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

M.Sc (APPLIED SCIENCE - COMPUTER TECHNOLOGY)

OF BHARATHIAR UNIVERSITY

Submitted by
S.Arulmurugan
0137Q0031

Under the supervision and guidance of
Mr. R.Dinesh, M.S [Computer Science - Wisconsin],
Assistant Professor,
Department of Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore – 641 006.

Department of Computer Science and Engineering

Kumaraguru College of Technology

(Affiliated to Bharathiar University)

Coimbatore – 641 006

APRIL 2003

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)
Department of Computer Science and Engineering

Coimbatore – 641006

CERTIFICATE

This is to certify that the project work entitled

"ONLINE MONITORING"

Done by

S.Arulmurugan
0137Q0031

Submitted in partial fulfillment of the requirement for the award of the degree of M.Sc (Applied science - Computer Technology) of Bharathiar University.

[illegible]
Professor and HOD

R. Dinesh
Internal Guide (21/4/20

Submitted to University Examination held on 10 – 5 - 03

R. Dinesh
Internal Examiner (10.5.2003)

[illegible]
External Examiner

DECLARTION

I hereby declare that the project entitled "*ONLINE MONITORING*" is successfully done at M/S Sovereign Infotech India Pvt. Ltd, Coimbatore and submitted to **Kumaraguru College of Technology**, Coimbatore afflicted to Bharathiar University as the project work of **M.Sc (APPLIED SCIENCE - COMPUTER TECHNOLGY)**, is a record of original work done by me during my period of study in Kumaraguru College of Technology, under the supervision and guidance of **Mr.R.Dinesh M.S.**[Computer Science - Wisconsin], Assistant Professor, **Kumaraguru College of Technology**, *Coimbatore*. And this project work has not formed the basis of award of any Degree / Diploma / Associate ship / Fellowship or similar title any candidate of any university

Name	:	S.Arulmurugan
Reg. No	:	0137Q0031
Signature:		[illegible]

SOVEREIGN INFOTECH INDIA PVT. LTD.,
Corporate Office
EA-503, PSG Step, PSG College of Technology, Coimbatore-641004. INDIA
Ph : 0422-2591873, 2578423/27/54, 98422 - 34870 Fax : 0422-2232944
E-mail : gvillage@vsnl.net / www.globalvillage-india.com

Ref : *Date :* 07-04-03

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr. S.ARULMURUGAN (Reg.No : 0137Q0031) doing M.Sc Applied Science Computer Technology in KUMARAGURU COLLEGE OF TECHNOLOGY, Coimbatore has successfully completed the project titled "ONLINE MONITORING". The duration of the project is December 2002 to March 2003.

Due to operational reasons, we are not in a position to give the software to the student.

We wish him all success.

For Sovereign Infotech India Pvt Ltd

[illegible]

P.Balakrishnan
Whole Time Director

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I would like to express my gratitude to our beloved Principal **Dr.K.K.Padmanabhan**, Ph.D., Kumaraguru College of Technology, Coimbatore, for his constant encouragement throughout my course.

I wish to thank **Dr. S.Thangaswamy**, Ph.D., Head, Department of Computer science, Kumaraguru College of Technology, Coimbatore, for his invaluable guidance and suggestions that encouraged me to complete this project successfully.

I admit my heart full thanks to my project guide and our course co-coordinator, **Mr. R.Dinesh**, M.S.[Computer Science - Wisconsin], Assistant Professor, Department of Computer Science, Kumaraguru College of Technology, Coimbatore, for being supportive throughout the tenure of my project.

My heartfelt thanks and gratitude to **Mr. D.Prabhu**, M.B.A., Director, Sovereign InfoTech India Pvt Limited, Coimbatore, for granting me permission to undertake the project work to their software concern at Coimbatore.

I extend my deep gratitude and thanks to **Mr. V.Santhakumar**, B.Tech., Project leader, Sovereign InfoTech India Pvt. Limited, Coimbatore, for extending valuable suggestions, support and motivation throughout the project work.

I also take this opportunity to extend my sense of gratitude to all the faculty members, non-teaching staffs of the Computer science Department, K.C.T, Coimbatore, for their guidance and co-operation rendered throughout my course.

SYNOPSIS

SYNOPSIS

"ONLINE MONITORING" is network administrator utility software, developed for M/S SOVEREIGN INFO TECH PVT Limited, Coimbatore. This Software takes snapshots of the client systems screen that are connected to a Local Area Network(LAN) and displays it on the server. Also it lists the active nodes present in the LAN. Provision for viewing the remote system desktop from the server is made. Mailing is also available for text communication between the client user and the administrator.

The main users of this software are System Administrators who wish to monitor the client activities at any instance of time. The product is developed using JAVA programming language.

CONTENTS

CONTENTS

INRODUCTION

1. INTRODUCTION

1.1. Project Overview

"**ONLINE MONITORING**" is developed using JAVA-RMI. The functions of this utility software are capturing the screen, text communication between the client user and the administrator.

The Main features of the product are as follows:

- Capture the client screen

The first feature of this product is capturing the client screen and to display them in the server. This software is to monitor the performance of the programmer by capturing their desktop so that we can view what the programmer is doing at that instant.

In this software we have two types of applications one is server side application and the other is client side application. The server side application requests the client side application to capture the desktop of the client by supplying the system name. Then the client side application captures the desktop at that instant of that particular system and stores it in the form of pixel definition, this pixel definition is then converted to integer array. This integer array is accessed by the server side application and each array value is converted to its original form of pixel definition. Using this pixel definition that particular desktop is displayed in the server terminal,

- Mailing

The Mail Program in the Client and the Server enables the users to communicate between them in the form of Text Messages. The administrator can create a list of users with a login name and password and send or receive messages. The administrator can also view the users who have received new messages. The user can also login into the tool to send or receive messages.

1.2. Organization Profile

The Rationale

The growth in Information Technology (IT) industry is phenomenal and there is no industry in the history of world that has grown as fast as IT, and within IT. It is the software and services sector, which is registering the highest growth rate. This is the guiding principle that let to the setting up of "Sovereign InfoTech India Pvt. Ltd."

Sovereign InfoTech India)P Ltd.

The main objective of the company's establishment is:

- To provide world class training this is relevant to the industry by associating with leading Institutes and Industries in India and adroad.
- To provide world class services at an affordable price without compromising Quality.
- To provide an opportunity to develop personal skills and knowledge so as to enable an individual to achieve what he/she is capable of achieving and contribute so that the society at large is benefited.

To achieve the above-mentioned objectives, Sovereign InfoTech India Pvt. Ltd., will provide:

- A wide range of software programmers in the latest technology, which is relevant to the industry.
- Partnering for joint ventures.
- Hardware Training and Technology and related services.
- Assist for placements in the Industry.

Project Training

Sovereign InfoTech India Pvt. Ltd. has trained more than 200 B.E/M.C.A/M.Sc/M.B.A from different colleges of Tamilnadu. The students are given throughout training in basic subjects followed by "on the job" training in project management, software quality etc., followed by "Real Time Projects" in the software development group. Thus the students get "Real Time Project" experience while undergoing the training itself.

The facilities

Sovereign InfoTech India Pvt. Ltd., is located at PSG STEP 1, PSG College of Technology campus, Coimbatore, which has got the highest reputation in drawing talents from across the globe. M/S Sovereign InfoTech India Pvt. Ltd., is having well-equipped computer labs with latest configuration machines and software and one machine-one student ratio is maintained throughout training period. A well stocked technical library caters to the students and every student has access to e-mail and Internet.

The Staff

The staff members have very high academic and professional qualification and many of them have industry experience. The Sovereign InfoTech India Pvt. Ltd., faculty at Coimbatore is selected after rigorous recruitment and selection process. They are highly qualified full time staff, formally trained to impart high quality training at various levels. They have rich experience in software development, training, consulting and research and development. Their involvement in consultancy, especially in software consultancy helps them handle the classes with real life experience.

Training Methodology

Understanding of concepts and skills development is given paramount importance at Sovereign InfoTech India Pvt. Ltd., For GUI based modules, online training is important for better understanding and retentively. Sovereign InfoTech India Pvt. Ltd., is providing training in software skills apart from providing software development services to leading Industries.

SYSTEM STUDY AND ANALYSIS

2. SYSTEM STUDY AND ANALYSIS

2.1. Existing System

, There is no such system which has all the functionality but as individual modules. The administrator has to switch over software for every function needed. It has also has the limitations of

- In screen capture the image captured is not sharp.
- Doesn't provide simple and efficient user interface.
- No sufficient interaction between the user and the software.

2.1.1. System Analysis

System analysis is an activity that encompasses most of the tasks collectively called as Computer Science Engineering. This is the most important step in a software project where we get a general idea about the needs of the end-users by having man-to-man conversation with them, and about the various conditions and restrictions that have to be taken care of while developing the software.

- Identify the user's need
- Evaluate the system concept for feasibility
- Perform economic and technical analysis
- Allocate functions to hardware, software, people, database and other system elements
- Establish cost and schedule constraints
- Create a system definition that forms the foundation for all subsequent engineering work

2.1.1.1. Identification of need

As a first step in the analysis of the system, the end-users of the proposed project were met to get first hand information regarding their needs and wants. Ideas from both the sides were exchanged in order to get a standard and satisfactory system.

2.1.2. Feasibility Study

Every project is feasible for given unlimited resources and finite time. Feasibility study is an evaluation of system proposed regarding its workability, impact on the organization, ability to meet user-needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study, before it is approved for development.

Feasibility and risk analysis are related in many ways. If a project risk is great the feasibility of producing software is reduced. In case of the wizard creation, the feasibility is analyzed and the system is checked for its workability and the impact on the organization and also the ability to meet user-needs and effective use of resources.

During feasibility analysis in this project, the following three primary areas of interest were considered very carefully. Such as:-

- Technical feasibility
- Economic feasibility
- Operational feasibility
- Social feasibility
- Management feasibility
- Time feasibility

2.1.2.1. Technical feasibility study

A study of function performance and constraint that may affect the ability to achieve an acceptable system. Technical feasibility centers on the existing system in system in which all activities are undertaken manually. This work will be sometime pending for a very long time as this is very manual, repetitive and highly non-technical way of doing the administrative task.

2.1.2.2. Economical Feasibility

An evaluation of development cost weighed against the ultimate income or benefit for the development system or product. Economic feasibility deals with the analysis of cost against benefits i.e., whether the benefits to be enjoyed due to the new system is worthy when compared for the cost to be spent on the system. Additional cost incurred in system construction, maintenance and mobilizing manpower to work on turnkey solution throws a very big challenge on the organization. Especially in the present scenario throws a very big challenge on the organization. Especially in the present scenario where the objective is towards centralization, reduce cost of software, hardware and cutting exponential growth of the size of the organization. Cost of the system is not very high when compared to the existing.

An evaluation of development cost (extra hardware and software needed) is weighed against the ultimate income or benefit derived from the developed system. Hence this project was economically justified for development in this organization.

2.1.2.3. Operational feasibility study

A determination of any infringement, violation or liability that could result from the development of the system. An evolution of alternative approaches to the development of the system.

A feasibility study is not warranted for system in which economic justification is obvious. Technical risk is low, few legal problems are expected, and no reasonable alternative needed for the project development.

1.2.4. Social Feasibility

Social feasibility is a determination of whether a proposed project will be acceptable to the people or not. This determination typically examines the probability of the project being accepted by the group directly affected by the proposed system change.

2.1.2.5. Management Feasibility

It is a determination of whether a proposed project will be acceptable to management. If management does not accept a project or gives a negligible support to it, the analyst will tend to view the project as a non-feasible one.

2.1.2.6. Time Feasibility

Time feasibility is a determination of whether a proposed project can be implemented fully within a stipulated time frame. If a project takes too much time it is likely to be rejected.

2.2. Proposed system

The aim of the proposed system is to monitor the client user activities in the software development lab and to manage them efficiently by the administrator The proposed system is a platform independent system that can run any type of platform, developed using Java RMI – Remote Method Invocation and MSACCESS for Database.

Proposed System Objectives:

- Provide simple & efficient user interface.
- Captures the client screen.
- Message communication between user and administrator.

After development the system will be put to various testing procedures. The testing procedures would ensure that the new system is error free from bugs. All the functional part of the system would be tested for defects. This would optimize the system in the usage of memory.

The processing schemes take the concept of time complexity by optimizing the loop structures and memory allocations to dynamic variables. This would ensure optimized usage of resources making the system more fast and reliable. The coding structure and the standard would ensure that there are no memory leaks and that the allocated memory is returned to the free store.

The proposed system is to have tool tip help to the menus, tool bar, item etc. this would ensure an improved way to access the functions of the systems. High lighting facility in the data display screens would ensure quick viewing of data.

Thus the proposed system is to take advantage of all the new technologies and ensures a smooth functioning.

2.3. Requirements on New System

2.3.1. Scope

- Efficiently monitor the clients connected to the network.
- Minimize the system administrator's desk time
- The system provides easy user-friendly interface to the users.
- User authentication is provided by username & password.
- Send message between user and administrator.

2.3.2. General Description

2.3.2.1. Product Functions

The following points would briefly explain the functions of the proposed system.

- Capture full screen/active window of client system.
- Message communication between user and administrator.

2.3.2.2 General Constraints

The general constraints regarding this software are:

- A time gap occurs between sending the request and displays the output.
- Whenever a new system is added to the network, its details should be entered.
- The server program installed in the client systems should be loaded at time of system startup so that they remain as a background process & accept the requests.
- Except the administrator no other person should access this program excluding chat.

2.3.2.3. Functional Requirements

Inputs:

The users of the system should be provided with a point & click interface, i.e., the users of the software mainly users mouse to provide inputs. Initially the user should authenticate themselves by means of a user name and password. Once the authentication process is over user is allowed to perform the required operations provided by the software. The only input for the server program (installed in client systems) is that the user should start the server so that it can accept incoming requests. The primary input is the IP address of the client system to be captured.

2.3.3.4. Performance Requirements

Security

Security should be implemented by means of a user name/password validation process. Only authorized users should be allowed to access the software.

Availability

Availability is the probability that a program is operating according to requirements at a given point of time. The availability is an indirect measure of the maintainability of software. The successful function of the software depends on the validity of the inputs given to it. If the data entered is not appropriate or data is missing the system should indicate possibility of an error.

Capacity

Capacity measures number of systems a software can access. The software should capture one user screen at a time.

Response Time

Response time is the time with in which a system identifies the instruction of the user and responds to it. The time required to capture the screen/process from the client system and to display it in the server should be approximately 5 seconds.

2.3.3. Design Constraints

2.3.3.1. Hardware Limitations

- A Pentium processor of 166Mhz speed(to make sure that application does not take too long to run)
- A random access memory of 64 MB.
- A mouse and keyboard
- LAN(Ethernet Interface)

2.3.3.2. User Interface and Screen Formats

It is required to maintain certain GUI standards during the development of this system. The user can either use a Mouse or a Keyboard to operate the product.

2.3.3.3. Operation required by the user

The users of the system should access the software after providing a user-name & password. The only operation required by the user is to provide inputs to the system. The user should provide valid inputs to the system.

2.4. User Characteristics

The system has been designed as a very easy to understand point-click interface system. The whole system is partitioned into two-Client side and Server side depending on the mode of operation of the system.

The main users of the system are system administrators. Since the product is user friendly the user need to possess only minimum data very knowledge. They should also have knowledge on basic networking concepts.

PROGRAMMING ENVIRONMENT

3. PROGRAMMING ENVIRONMENT

3.1 HARDWARE SPECIFICATION

System	IBM compatible PC
Processor	Intel Pentium II
Hard Disk Drive	20 GB
RAM	64 MB
Monitor	14" SVGA Monitor
Keyboard	Samsung 104
Mouse	Logitech

3.2 SOFTWARE SPECIFICATION

Platform	Microsoft Windows or Any OS
Software	JAVA (JDK 1.3.1)
Packages	JavaRMI-Remote Method Invocation
Database	Ms-Access (Backend)

3.3 ABOUT THE SOFTWARE

3.3.1 JAVA-Features

Simple

Java was designed to make it much easier to write bug free code. According to Sun's Bill Joy, shifting c code has an average one bug per 55 lines of code. The most important part of helping programmers to write bug free code is keeping the language simple.

Java has the bare bones functionality needed to implement its rich features set it does not adds syntactic sugar or unnecessary features despite its simplicity Java has considerably more functionality than C. Primarily because of the large class library because Java is simple it is easy to read and write. Obfuscated Java isn't as common as Obfuscated C. There aren't a large special cases or tricks that will confuse beginners about half of the bugs in c and c++ programs are related to memory allocation and reallocation therefore the 2 important addition Java makes to providing hug free code is automatic memory allocation and reallocation. The C library memory allocation functions Malloc() and Free() are gone as C++'s destructors.

Java is an excellent teaching language and excellent choice with which to learn programming the language is small so it s easy to become fluent. The language is interpreted so the compile run link cycle is much shorter. The run time environment provides automatic memory allocation and garbage collection so there's less for the programmer to think about. Java is object oriented unlike basic so the beginning programmer doesn't have unlearned bad programming habits when moving into real world projects. Finally it is difficult (if not quite impossible) to write a Java program that twill crash our system something that u cant say about any other language.

Object Oriented

Object oriented programming is the catch phrase in 1990's.although object oriented programming has been around in form or another since the similar language was invented in the e1960's. Its really big end to take hold in modem GUI environment like Windows, Mac, Motif In object oriented programming data is represented by objects. Objects have two sections Fields (Instance Variables) and Methods. Fields tell u what an object is, Methods tell u what an object tells you these fields and methods are closely tied to the object's real world characteristics and behavior. When a program is run, messages are passed back and forth between objects. When an object receives message response accordingly as defined by the methods.

Object oriented programming is alleged to have a number of advantages including:

- Simpler easier to learn program's
- More efficient the use of code
- Faster time to market
- Most robust Error

In practice object oriented programs have been just as slow expensive and buggy as traditional non-object oriented programs. In large part this because the most popular object oriented language is C++. C++ is a complex difficult language that shares all the frustration of c while sharing none of C's efficiency. It is possible in practice to write clean, easy to red java code An c++ this almost unheard off outside of programming, textbooks.

Security

Java provides a "fire wall" between a network application and the user's computer. Using a Java compatible Web browser, any user can safely download Java applets without fear of viral infection or malicious intent. Java achieves this protection by confining a Java program to the Java execution environment and not allowing it access to other parts of the computer. The ability to download applets with confidence that no security will be breached is considered by many to be the single most important aspect of Java.

Portability

Many computers and operating systems are in use throughout the world - and many are connected to the Internet. For programs to be dynamically downloaded to all the various type of platforms connected to the Internet, some means of generating portable executable code is needed.

Bytecode

The output of a Java compiler is not executable code. Rather, it is byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called Java Virtual Machine (JVM). JVM is an interpreter for bytecode. JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it.

Distributed

Java is designed for distributed environment of the Internet, because it handles TCP/IP protocols. Infact accessing a resource using a URL is not much different from accessing the file. The original version of Java included that features for intra address-space messaging. This allowed objects on two different computers to execute procedures remotely.

Platform Independent

Java was designed to not only be cross platform in source form like C, but also in compiled binary form. Since this is frankly impossible across processor architectures java is compiled to an intermediate form called byte code. A java program never really executes natively on the host machine. Rather a special native program called a java language interpreter and some of the library routines. Even the compiler is written in java. The byte code's are precisely defined and remain the same on all platforms.

The second important part of making java cross platform is the elimination of undefined or architecture dependent constructs. Integers are always four bytes long and floating point variables follow the IEEE 754 standard for computer arithmetic exactly.

How ever virtual machine itself and some parts of the class library must be written in native port. These are not always as easy or quick to port as pure java programs. This is why for example there's not yet a version of java 1.2 for the Mac.

High Performance

Java byte codes can be implemented on the fly to code those rivals C++ in speed using a "just in time compiler ". Several companies are also working on native machine architecture compilers or java. These will produce executable code that does not require a separate interpreter and that is distinguishable in speed for C++. When you'll never get that last ounce of speed out of a java programming that you might be able to write from C the results will be suitable for all but the most demanding applications on May 1999 the fastest VM. IBM's java 1.1 VM for Windows is very close to C++ on CPU intense operations that don't involve a lot of disks I/O or GUI work: C++ is itself only a few percent slower than C on CPU intensive operations. It is certainly possible to write large programs in java. The hot browser, the java Workshop integrated development environment and the java compiler are large programs that are written certainly in java.

Dynamically Linked

Java does not have an explicitly linked phase. Java source code is divide into java files roughly one per each class in your program. The compiler compiles these into class files containing white code. Each java file generally produces exactly one class file.

The compiler searches the current directory and directories specified in the class path environment variable to find other classes explicitly reference by name in each source code file. The file you are compiling depends on other non-compiled files the compiler will try to find them and compile them as well. The compiler is quite smart and can handle circular dependencies as well as methods that are used before they are declared. It also can determine whether a source code file has just changed since the latest line it was compiled.

More importantly classes that where unknown to a program when it was compiled can still be loaded quick at run time. For example a web browser can load applets of referring classes that has never seen before without recompilation. Further more java class fields and to be quite small a few kilobytes at most. It is not necessary to link large run time libraries to produce an executable. Instead the necessary classes are loaded from the user's class path.

Garbage Collector

You do not need to explicitly allocate or de-allocate memory in java, memories allocated, as needed both on the stack and reclaim by the garbage collector when it is no longer needed. There's no malloc or free methods. There are constructors and these allocate memory on heat but this is transparent to the programmer.

Most java virtual machines use an inefficient mark and sweep garbage collector. Java is a platform is loosely to find computer industry buzz word that typically means some combination of hardware in system software that will mostly run all the same software. For instance power Mac's running system 7.5 would be one platform, DEC alphas running Windows NT would be another, The another problem with distributing executable programs from web pages. Computer programs are very closely tied tc the specified hardware and operating system they run. Windows will not run on machine. There fore major commercial applications like Microsoft word or Netscape have to be written almost independently for all the different platforms they run on. Netscape one of the most cross platform of major applications and still only runs on a minority of platforms.

Java solves the problem of platform independence by using byte code. The java compiler does not produce native executable code for a particular machine like C compiler would. Instead it produces the special called byte code java byte code 'written hex decimal, byte-by-byte looks like this:CAFEBABE0003 002D0022003D4FFAA0118003 This looks a lot like machine language. But unlike machine language java byte code is exactly the sane on every platform. This byte code fragment means the same thing on a Solaris workstation as it does on a Mac power book. Java programs that have been compiled to byte code still need an interpreter to execute them on any given platform. The interpreter needs a byte code and translate it into the native language of the host machine on the fly. The most common such interpreter is Sun's program java. Since the native libraries need to be ported to get java run on new computer or operating system. The rest of the run time environment including the compiler and most of the class libraries are written in java. All these pieces java compiler and java interpreter the java programming language and more are collectively refer to as java.

Thread

In computer programming, a thread is placeholder information associated with a single use of a program that can handle multiple concurrent users. From the program's point-of-view, a thread is the information needed to serve one individual user or a particular service request. If multiple users are using the program or concurrent requests from other programs occur, a thread is created and maintained for each of them. The thread allows a program to know which user is being served as the program alternately gets re-entered on behalf of different users. (One way thread information is kept by storing it in a special data area and putting the address of that data area in a register. The operating system always saves the contents of the register when the program is interrupted and restores it when it gives the program control again.)

Multithreading

It is easy to confuse multithreading with multitasking or multiprogramming, which are somewhat different ideas. Multithreading and are similar and are often confused. Multithreading is the management of multiple concurrent uses of the same program. Most operating systems and modern computer languages also support Multithreading. Today's computers can only execute one program instruction at a time, but because they operate so fast, they appear to run many programs and serve many users simultaneously.

The computer operating system (for example, Windows 95) gives each program a "turn" at running, then requires it to wait while another program gets a turn. Each of these programs is viewed by the operating stem as a "task" for which certain resources are identified and kept track of the operating system manages each application program in your PC system (spreadsheet, word processor, Web browser) as a separate task and lets you look at and control items on a task list.

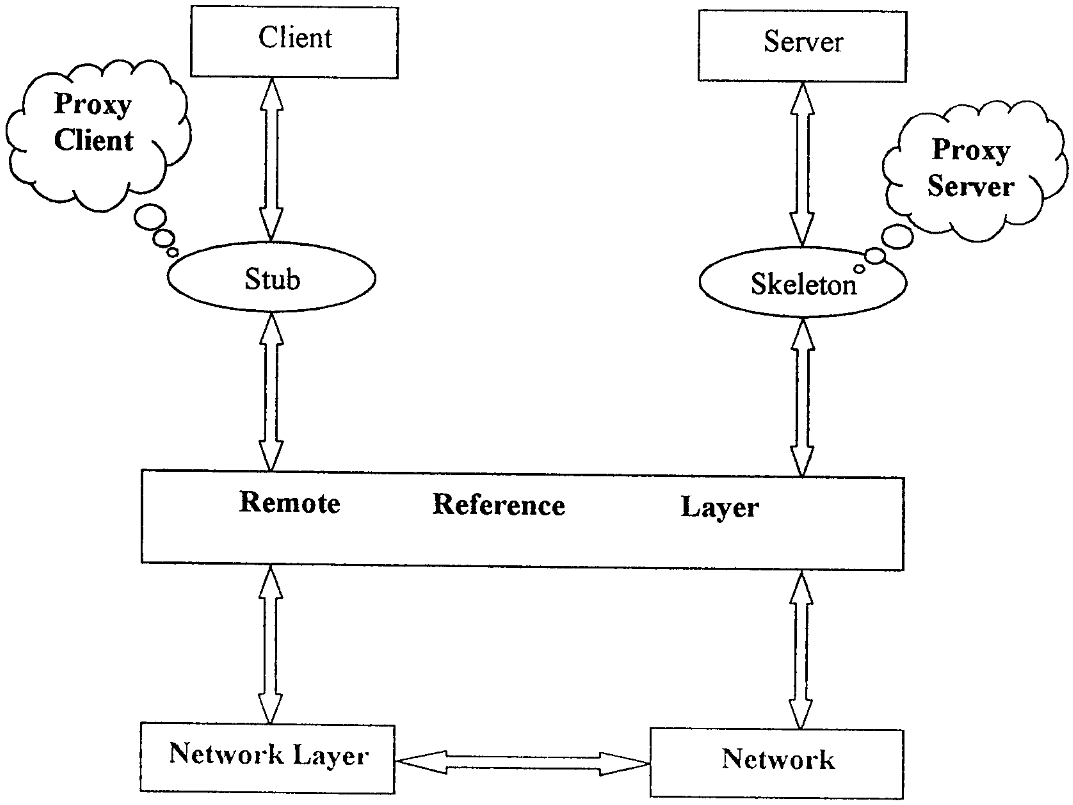
Meanwhile, other concurrent uses of the program are maintained on other threads. Most of today's operating systems provide support for both multitasking and multithreading. They also allow multithreading within program processes so that the system is saved the overhead of creating a new process for each thread.

Multithreading is the ability of a program or an operating system process to manage its use by more than one user at a time and to even manage multiple requests by the same user without having to have multiple copies of the programming running in the computer. Each user request for a program or system service (and here a user can also be another program) is kept track of as a thread with a separate identity. As programs work on behalf of the initial request for that thread and are interrupted by other requests, the status of work on behalf of that thread is kept track of until the work is completed.

3.3.2 RMI -Remote Method Invocation

Remote Method Invocation (RMI) has several new enhancements. Remote Object Activation introduces support for persistent references to remote objects and automatic object activation via these references. RMI over a secure transport can be supported using custom socket factories. Minor API Enhancements allow the following: unexporting a remote object, obtaining the stub for an object implementation, obtaining a local object . implementation from a stub, and exporting an object on a specific port.

The RMI interface architectural diagram.



Package java.rmi Description

RMI is Remote Method Invocation. It is a mechanism that enables an object on one Java virtual machine to invoke methods on an object in another Java virtual machine. Any object that can be invoked this way must implement the Remote interface. When such an object is invoked, its arguments are "marshaled" and sent from the local virtual machine to the remote one, where the arguments are "un marshaled." When the method terminates, the results are marshaled from the remote machine and sent to the caller's virtual machine. If the method invocation results in an exception being thrown, the exception is indicated to caller.

Java TM Remote Method Invocation (RMI) is a distributed object model for the Java language that retains the semantics of the Java object model making distributed objects easy to implement and to use. The system combines aspects of the Modula-3 Network Objects system and spring's subcontract and includes some novel features made possible by java.

Stubs and Skeletons

RMI uses a standard mechanism (employed in RPC systems) for communicating with remote objects: stubs and skeletons. A stub for a remote object acts as a client's local representative or proxy for the remote object. The caller invokes a method on the local stub, which is responsible for carrying out the method call on the remote object. In RMI, a stub for a remote object implements the same set of remote interfaces that a remote object implements. When a stub's method is invoked, it does the following:

- Initiates a connection with the remote VM containing the remote object.
- Marshals (writes and transmits) the parameters to the remote VM
- Waits for the result of the method invocation
- Un marshals (reads) the return value or exception returned
- Returns the value to the caller

The stub hides the serialization of parameters and the network level communication in order to present a simple invocation mechanism to the caller.

In the remote VM, each remote object may have a corresponding skeleton (in JDKI.2-only environments, skeletons are not required). The skeleton is responsible for dispatching the call to the actual remote object implementation. When a skeleton receives an incoming method invocation it does the following:

- Un marshal (reads) the parameters for the remote method
- Invokes the method) the result (return value or exception) to the caller
- In JDK1.2 and additional stub protocol was introduced that eliminates the need for skeletons in JDKI.2-only environments. Instead, generic code is used to carry out on the actual remote object implementation .
- Marshals (writes and transmits the duties performed by skeletons in JDK1.1 Stubs and skeletons are generated by the rmic compiler.

iava.rmi.server.RemoteServer

The method unexportObject has been moved to the two subclasses of Remote Server, java. rmi.scrver. Unicast RemoteObject and java.rmi.activation. Actvatable.

java.rmi.server.SocketType

The class Socket Type has been removed and replaced with a more flexible mechanism for using custom socket factories on a per remote-object basis. We have introduced two new interfaces, java.rmi.server.RMI ClientSocketfactory and java. rmi.server.RMIServerSocketFactory, that can be specified when a java. rmi.scrver.UnicastRemoteObject or java.rmi.activation. Activatable object is constructed/exported. The RMI runtime uses the specified RMIServerSocketFactory to create a ServerSocket to acc6pt incoming calls for the remote object. The remote object's reference contains a reference to an RMIChentSocketFactory which is downloaded to clients, when transmitted in an RMI call, and is used to make connections to the remote object for a remote method call.

The reason we replaced the Socket Type abstraction with the custom client and server socket factories is to allow a client socket factory to be downloaded into applets so that no client-side socket factory installation is required. The constructors and export Object methods of java.rmi.scrver. Unicast RemoteObject and java rmi.activation.Actlvzitable that previously took a Socket Type as one of their arguments have been modified to take the custom socket factories as parameters instead.

java. rmi server. RMISocket Factory

The java.rmi.server.RMISocketFactory class now implements jva.rmi.server.RMIClientSocketFactory java.rmi.server.RMIServerSocketFactory. The create Socket and createServerSocket methods that took a Java.rmi.server. Socket Typeas a parameter have been removed.

java.rmi.server.UnicastRemoteObject

The UnicastRemoteObject class has several modified constructors and exportObject methods, and a new unexportObject method. The constructors and exportObject methods have been changed to use a java. rmi.server. RMIClientSocketFactory and java.rmi.server.RMIServerSocketFactory as arguments instead of a java.rmi.server. SocketType. The unexportObject method was moved from java.rmi.server.RemoteServer to UnicastRemoteObject

Remote Interfaces

A remote interfac'e can now extend an interface that does not extend java.rmi.Remote as long as all of the methods throw a lcgal remote exception type (see Remote Exceptions, next).

Remote Exceptions

A remote interface method can alternatively use any of the superclasses of java.rmi.RemoteException, such as java.io.IOException or java. lang.Exception, instead of RemoteException in the throws clause of the method.

java.rmi.registory.Locate Registry

Two new methods, createRegistry and getRegistry, have been added to the LocateRegistry class. The createRegistry method allows you to create a remote object registry that uses custom client and server socket factories for communication with that registry. The get Registry method returns a locally created reference to a remote object registry that uses a specified RMIClientSocket Factory to obtain sockets to communicate with that remote registry.

java.rmi.RMISecurityManager

The RMISecurtyManager has been updated to work with the new JDKI.3 security model. Most of the RMISecurityManager methods are no longer overridden since the default behavior of the methods of java.lang. SecurityManager, which it extends, is appropriate for the RMISecurityManager. Please note that if you do use the RMISecurityManager, you will need to use a policy file that lists the permissions that are granted to code from various codebases.

java.rmi.server.RMIClassLoader

The RMIClassLoader class has two new methods, getClassAnnotation and a load Class method that takes a codebase path and a class name as arguments.The getClassAnnotation method returns a string containing the class's codebase path that RMI transmits with the class descriptor in remote method calls, which include an instance of that class in a parameter, return value, or exception.

java.rmi.activation.Activatable

The Activatable class has several new constructorsand exportObject methods and a new unexportObject method.

The constructor export Object methods used java. rmi.server. RMIClientSocketFactory, java.rmi.server.RMIServerSocketFactory as arguments and facilitate custom communication with activatable objects. The unexportObject method was moved from java.rmi.server.RemoteServer to Activateable. Also so note that java.security.CodeSources is no longer used to represent the location of the activatable object's class. Instead, a java.lang.string is used to represent a codebase path from where the class bytecodes can be loaded. A codebase path is a string of URLs separated by a space. Due to this change, the constructors and exportObject methods now take a string instead of a CodeSource for the codebase argument.

java.rmi.activation.ActivationGroupDesc

Since we now use a java.lang.String to represent the location, a codebase path, of the activatable object's class, the ActivationGroupDesc constructors that previously took a java.security.CodeSource as the codebase argument, now takes a String instead. The getCodeSource method has been renamed to getlocation for consistency. ActivationGroupDesc also supports a new scheme for specifying the properties and environment in which a group executes.

java.rmi.server.ObjID

We have introduced a new property java. rmi.server. randomIDs that, if true, ensures that an ObjID (contained in a remote object reference) contains a cryptographically secure random number. The property defaults too false. If the property is set to true, you will experience several seconds of delay when the first ObjID is generated while the secure random number generator determines its seed.

java.rmi.server.useLocalHostname property

RMI now uses an IP address for the local host if the java.rmi.server.hostname property is not specified and a fully qualified domain name for the localhost cannot be obtained. In order to force RMI to default to the hostname, you need to set the java.rmi.server.useLocalHostname property to true. The useLocalHostname property defaults to false.

java.rmi.activation.server-RemoteObject

The method to Impl has been removed since the method cannot be made secure without allowing too many restrictions on its use. The accessor method getRef has been added. It returns the remote reference type, a java.rmi.server.RemoteRef for the remote object.

java.rmi.server.RemoteServer

The method unexportObject takes an additional parameter force and returns a Boolean to indicate whether the operation was successful.

java.rmi.server.UnicastRemoteObject

Two forms of the export Object method now return Remote instead of RemoteStub since the former allows more flexibility in future implementations of RMI.

java.rmi. activation.ActivationDesc

Two constructors have been added which take an additional parameter, restart, to indicate the restart mode for the object. The restart mode, if set, will cause the object to be restarted when the RMI activation daemon starts; otherwise, the object is simply activated on-demand .An additional method getRestartMode returns the restart mode for the object.

java.rmi.activation.Activatable

An additional constructor that takes a restart parameter. The register method now returns Remote instead of RemoteStub since the former allows more flexibility in future implementations of RMI. The inactive method now returns a Boolean that indicates whether the object was deactivated successfully. An object can only be deactivated if it has no pending or executing calls.
An additional exportObject method was added that is similar to the first but takes another parameter restart that indicates the restart mode of the object.

java.rmi.activation.ActivationSystem

An additional method shutdown has been added to provide a means for graceful shutdown of the activation system. Also, a new option, -stop has been added to rmid that invokes the shutdown method of the ActivationSystem running on the default or specified port.

3.3.3 History of TCP/IP

The TCP/IP protocols were first adopted by the DARPA Defense Advanced Research Projects Agency (DARPA) as part of a project to connect primary government funded research institutions in a computer network. Around the same time at the University of California at Berkeley, the BSD (Berkeley Systems Distribution) version of UNIX was widely distributed to university computer science departments. DARPA contracted to have the TCP/IP protocols merged into the Berkeley BSD operating system. Then, DARPA required anyone hooking up to the Internet to run the TCP/IP protocols.

Because of DARPA, the TCP/IP protocol source code was widely distributed as part of BSD. All TCP/IP source code leveraged from the BSD source~ had to contain the University of California at Berkeley copyright. From this leveraged source code, vendors ported the TCP/IP protocols to most mainframe and mini-computers.

Then as the embedded industry grew, RTOS vendors and separate protocol stack vendors ported the TCP/IP sources from BSD to many embedded environments. Most of the products using TCP/IP today contain much of the unaltered code from Berkeley and much of it still carries the BSD Copyright.

Network

The network layer encompasses the Internet domain knowledge. It contains the routing protocols, and it understands the Internet addressing scheme. The IP addressing mechanism must be converted into Physical (MAC) addresses before the link and physical layers can transmit the data.

11B**Internet Protocol)IP**

The IP (Internet Protocol) is the part of the stack known as the network layer. The network layer handles routing of packets across network interfaces. Domain naming and address management are considered to be part of this layer as well. IP also includes a mechanism for fragmentation and re-assembly of packets that exceed the link layer's maximum transmission unit (MTU).

IP Addressing

IP addresses are used to insure packets are routed to the correct destination. Also, IP addresses can be coded as broadcast and multicast addresses. It is the convention to use the dotted decimal notation to describe IP addresses. It is generally more convenient to express net masks as hex numbers. The class of the address determines the number of machines on a subnet and hence the type of sub netting. It is not necessary that the subnet. Be divided on a byte boundary. Class A and B addresses are not used very much any more because they allow too many hosts on a subnet. Generally, Class C or subdivided class B addressing is used. The subnet mask defines the type of addressing.

3.3.4. MICROSOFT ACCESS 7.0

About MS Access 7.0

Microsoft Access 7.0 is the DBMS developed by Microsoft Corporation for small office management purpose. MS Access is the part of Office 2000 an office automation suite for small offices, containing MS-Word document writer, MS-Excel a spreadsheet, MS-Power point a presentation designer.

MS-Access is a very easy environment to create and manipulate databases. It provides GUI interactivity to access database. It provide the query builder, from designer, report generator, macro writer and module writer to create retrieve and make hardcopies of required information stored in database.

Components of MS-Access 7.0

Tables

Tables are the combination of rows and columns used to store the data in database as the columns as fields and the row as record's Access allows creating tables very easily. We can easily specify the type of field and incorporate some validations.

SQL Commands

Create: This command is used to create a new define table structure

Update: This commands used to modify the contents of the table, using this command a particular set of records can also be updated.

Insert: This command saves, is used to insert new records to the user table.

Delete: This delete command is used to delete a data from the table.

Retrieve: This command is used to retrieve values from the table. It set relational databases with a single command. Selected output can be directed to another table.

SYSTEM DESIGN AND DEVELOPMENT

4. SYSTEM DESIGN AND DEVELOPMENT

System design is a solution, a "how to" approach to the creation of a new system. This important phase is composed of several steps. It provides the understanding and procedural aspects details necessary for implementing the system recommended in the feasibility study.

Emphasis is on translating the performance requirements into design specifications. Design goes through logical and physical stages of development. System design is the development of a blueprint of a computer system solution to a problem that has the same components and interrelationships among the components as the original problem. System design has the following properties

- Schedules design activities
- Work with user to determine the various inputs to the system
- Plans how data will flow through the system
- Designs required outputs
- Program specification

The design of a system is essentially a blue print, or a plan for a solution for the system. Compound of a system can itself be considered a system, with its components. The design process for a software system has two levels.

- System Design
- Logical Design

System Design

Here the focus is on detecting which modules are needed for the system, the specification of these modules, and how the modules should be interconnected.

Logical Design

Here the internal design of the modules the specification of the module can be satisfied is decided upon. This also knows as detailed design.

4.1. Input Design

Input Design is the most important part of the overall system design, which requires very careful attention. Often the collection of input data is the most expensive part of the system. Many errors may occur during this phase of the design. So to make the system study, the inputs given by the user is strictly validated before making a manipulation with it. Thus, by validation it is possible to

- Produce an effective method of input.
- Achieve the highest possible level of accuracy
- Ensure that input is acceptable to and understood by the user staff.

Input design is concentrated on estimating what the inputs are and how often they are to be arranged on the input screen, how frequently the data are to be collected. The Online Monitoring provides many user-friendly features that help the user to interact with the system easily. The input screens are designed in such a manner that avoids confusion and guide the user in correct track. A thorough study has been made on the type and how the input form is to be designed. Some inputs from user may cause severe errors and is strictly validated. This software provides a point & click to its users. The layout of the input screen is also taken into account. A very good look and feel is provided through the organized arrangement of controls such as menus, edit fields, dialog boxes, buttons etc.

Types of inputs

The nature of the input data is determined from the beginning of the input design. The main input types can be categorized as,

- External
- Internal
- Operational
- Computerized
- Interactive

Input screen for the Online Monitoring are very simple and user-friendly. Users are allowed to access the software only after a user authentication process. If irrelevant data is entered message screens are displayed.

4.2. Output Design

Output design generally refers to the results generated by the system. For many end-users, output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application.

The objective of a system finds its shape in terms of the output. The analysis of the objective of a system leads to determination of outputs. The most common type of outputs is screen displays.

The outputs also vary in terms of their contents, frequency, timing and format. The users of the output, its purpose and sequence of details to be printed are all considered. If the outputs are inadequate in any way, the system itself is inadequate.

The basic requirements of output are that it should be accurate, timely and appropriate. When designing output, the system analyst must accomplish things like, determining what information to be present, whether to display or print the information, select output medium and to decide how to distribute the output to intended recipients.

The types of outputs are:

- External Outputs
- Internal Outputs
- Interactive outputs

External Outputs

These are the type of outputs, whose destination will be outside the organization and which require special attention as they project the image of the organization.

Internal Outputs

These are the type of outputs, whose destination is within the organization. It is to be carefully designed, as they are the user's main interface with the system.

Interactive outputs

These are the type of outputs, which the user uses in communicating directly with the computer.

4.3. Module Design

A module is defined as a collection of program statements with four basic attributes: input and output, function, and internal data. Modular systems consist of well-defined, manageable units with well-defined interfaces among the units. Desirable properties of a modular system include:

- Each processing abstraction is a will-defined subsystem that is potentially useful in other applications.
- Each function in each abstraction has a single, well-defined purpose.
- Each function manipulates no more than one major data structures.
- Functions share global data selectively. It is easy to identify all routines that share a major data structure.
- Functions that manipulate instances of abstract data types are encapsulated with the data structure being manipulated.

Modularity enhances design clarity, which in turn eases implementation, debugging, testing, documentation, and maintenance of the software product. The Online Monitoring project is divided into two main modules based on their functionality

- Remote Desktop Monitoring
- Mailing

Remote desktop monitoring

In this module the administrator is allowed to select any one of the system in the LAN and can select the desktop of the selected specific system. This gives the desktop of any active system and can be refreshed to get the next frame with a little time slot allocated. The desk top of the active system selected is grabbled by selecting the screen using the pixels. This can be done only for a single system at a time. It consists of two sub modules:

- Server side module
- Client side module

The connection establishment between the client side and server side modules is done by the methods available in java.rmi PACKAGE

Server-side module

This server side module basically has three sub modules in which each module has its own importance. The first module is used to check whether the user is authorized or not. The second module displays the nodes connected in the LAN. The third module displays the screen captured by the second module.

- Authentication
- Node Collection
- Screen Capture

Authentication

This is the first module used to check whether user is an authorized person or not. That is it Just checks whether the given username and password is valid or not. After giving the required details and when we click the ENTER button the control is transferred to the next module which is described below. The other button we have is the EXIT button. This buttons just quits out of the software. The Abstract Window Toolkit (AWT) supports Graphical User Interface (GUI) programming.

Node Collection

This module shows the active nodes connected in the networks that we could select a node whose screen or desktop is to be captured. After selecting a node & when we click the CAPTURE button, the current screen of the selected node will be captured and send it to the server and display the captured screen. The active nodes of the network are displayed using the methods present in the package called JAVA.NET PACKAGE.

Screen Capture

This module shows the captured screen of the intended node at the particular instant connected in LAN. The desktop of remote screen is captured pixel by pixel and send to the server. In the server all the collected pixels are grouped and displayed. The necessary frame to display the captured content of the remote desktop is developed by using the methods presents in JAVA.AWT PACKAGE.

Client side Modules

This module basically has two sub modules in which each method has its own importance. The first module is an interface to declare all the necessary methods used in the Remote Desktop Monitoring. The second module is define all methods that is presents in the interface by implementing the interface method.

- Interface
- Implementation

Interface

In Client side we use three methods which stays in client node always and sends the captured screen whenever the server request's the client side application to capture the screen of the node. In this we use three methods which capture the screen convert the pixel definition to integer format store the integer format in the array and sends to server whenever the Required. The three methods are

- getRemoteDeskTopWidth()
- getRemoteDeskTopHeight()
- getCapturedImagePixels()

Implementation

The implementation module defines the definitions which are declared in the interface module. The methods in first module are to get the client screen height, width in coordinate forms and to capture the screen of client and store the captured screen in the form pixel definition obtained from the display buffer and then to convert this pixel definition into its appropriate integer format, store it in an integer array. This array is sent to server whenever it is requested.

We use another file of Java named rmiregistry. rmiregistry is started as the user login the system and the system name is bounded in registry. So whenever the server application is executed giving the required system name as input, the server checks the rmiregistry for that particular system name. If the system name is found in rmiregistry then server communicates with the client to send the pixel definition array in the integer format. This is done using rmiregistry.

Mailing

Mailing module provides message communication between the client user and the administrator. This module tool provides the functionality of chatting. This module consists of two sub modules.

- Server side module
- Client side module

Server side Module

This server side module contains eight methods that are presents in the server machine. These methods are used by the administrator for the following purposes.

- Creating a user
- Deleting a user
- Changing user password
- View the user list
- View the message user list
- To compose mail
- To see inbox
- To display help messages

Creating a user

The Administrator can create a new user by providing UserName and PassWord by using the method CreateUser().

Deleting a user

The Administrator can delete some or all users from the user lists which are created by the administrator by using the method DeleteUser().

Changing user password

This option is used to change the existing passwords of the users by using the method ChangePwd().

View user list:

The Administrator can view the list of users which he has created by using the method ViewUser().

View message user list

The administrator can view the list of users who have received mails by using'the method ViewMessageUser ().

To compose mail

The administrator can compose mail to the user by using the method ComposeMail().

To see inbox

This option is used to see all the incoming mails by using the method InBox().

Display help messages

Providing the necessary help messages to the administrator for effective usage of mail server application.

Client_side Module

This client side module contains eight methods that are presents in the server machine. These methods are used by the client user and controlled by the administrator for the following purposes.

- To SignIn
- To see InBox
- To Compose mail

To SignIn:

This option is used to enter in to client application to send or receive mails. This option works only when the administrator has already created a login name or ID for him. If the user who has signed-in is a valid user, the name of that user is displayed on the home screen.

To see inbox

This option is used to list the users whose send the mails to the administrator by using the method ClientInBox().

To compose mail

The client user can compose mail to the user by using the method ClientComposeMail().

4.4. Database Design

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and effectively. The organization of data in the database helps to treat data as an organizational resource and as a integrated as a whole. A database is designed to meet the overall goal of the project and to store the information into separate database table. The tables are similar to ordinary files. Each record is stored in one row of these tables. Each Column of the table represents one particular field of the record. The database is normalized.

The design of the database forms is the nucleus of any application program. So the design of database always requires careful data analysis and scheme design. The Online Monitoring uses MSACCESS database named Mail. This database is only used for Mailing module of this project. It consisting of three tables named as follows.

- ServerMailInfo
- ClientMailInfo
- UserDetails

ServerMailInfo

This table contains seven fields that are:

FromUser - The mail come from which user

FromHost - The mail come from which host

ToUser - The mail go to which user

ToHost - The mail go to which host

Subject – Subject of the mail

FileName – Name of the mail content

Date_Time – Date and Time of the mail send

ClientMailInfo

This table contains seven fields that are:

FromUser - The mail come from which user

FromHost - The mail come from which host

ToUser - The mail go to which user

ToHost - The mail go to which host

Subject – Subject of the mail

FileName – Name of the mail content

Date_Time – Date and Time of the mail send

UserDetails

This table contains two fields that are:

UserName – Name of the client user

PassWord - PassWord to user

SYSTEM TESTING AND IMPLEMENTATION

5. SYSTEM TESTING AND MPLEMENTATION

5.1. System Testing

System testing is the critical element of software quality assurance and represents the ultimate review of specification, design and coding. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved testing includes verification of the basic logic of each program and verification that the entire system works properly. Test case design focuses on a set of techniques for the creation of test cases that meet overall testing objectives. The logical design and the physical design should be thoroughly and continually examined on paper to ensure that they will work when implemented.

When the programmers have tested each program with the test data designed by them, and have verified that these programs link together in the way specified in the computer run chart to produce the output specified in the program suite specification, the complete system and its environment must be tested to the satisfaction of the system analyst and the user.

Objectives of Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has high probability of finding an as yet undiscovered error. A successful test is one that uncovers an as yet undiscovered error.

Testing demonstrates that software functions work according to specifications. In addition, data collected from testing provides a food indication of software reliability and some indication of software quality as a whole. Testing results in the deduction in the number of errors. Critical modules are tested as early as possible.

Software testing for ONLINE MONITORING has been done during the pre-implementation stage using various software testing strategies and they are discussed below.

Testing Methods

Unit Testing

Unit testing was performed in order to check whether the developed programs function as specified. Unit testing is mainly used to test the smallest part (i.e.) module of the software design. Using the detailed design descriptions as the guide, important control paths are tested to uncover errors. Unit testing comprises the set of tests performed by the programmer prior to integration of the unit into a larger system.

A program unit is usually small enough that the programmer who developed it can test it in greater detail, and certainly in greater detail, than will be possible when the unit is integrated into an evolving software product.

There are four categories of tests that a programmer will typically perform on a program unit.

Functional tests

It involves exercising that the mode with nominal input values for which the expected results are known, as well as boundary values and special values.

Performance tests

These determine the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

Stress tests

These are those tests designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

Structure tests

These are concerned with exercising the internal logic of a program and traversing particular execution paths. Some refer collectively to function, performance and stress testing as black box testing while structure testing is referred to as white box or glass box testing. The major activities in structural testing are deciding which paths to exercise, deriving test data to exercise those path, determining the tests coverage criterion to be used, executing the test cases and measuring the test coverage achieved when the test coverage's achieved when the test cases are applied.

All modules of the newly developed system, as soon as they are completed, have been tested and errors found were corrected. Functional test has been carried out on all the modules to determine whether they produce expected results. Structure test was conducted in order to make sure that the internal logic of the modules is free of errors.

Sub - System Testing

This phase involves testing collection of modules, which have been integrated into sub-system. Sub-system may be independently designed and implemented. The most common problems that arise in large software system are sub-system interface mismatches. The sub-system process should therefore concentrate on the detection of interface errors by rigorously exercising these interfaces.

The modules available in the server-side alone are considered as a subsystem. Similarly in the case of client side also. Each and every module in the respective subsystem was integrated and their interfaces were tested.

The server-side and the client-side are treated as two separate sub-systems. These subsystems and their interface were also tested in this stage.

System testing

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

The two subsystems, namely, server-side and client-side, were integrated to form the whole system. The communication between these two sub-systems forms the major interface between them. This communication interface was tested for errors. The entire system was tested in this stage.

Interface testing

Interface testing takes place when modules or sub-systems are integrated to create larger systems. Each module or sub-system has a defined interface, which is called by other program components. The objective of this testing is to detect faults which may have been introduced into the system because of interface errors or invalid assumptions about interfaces. There are different types of interface between program components and consequently different types of interface error can occur.

Parameter interfaces: these are interfaces where data or sometimes function references are passed from one component to another.

Shared memory interfaces: these are interfaces where block of memory is shared between sub-systems and retrieved from there by other sub-system.

Procedural interfaces: these are interfaces where one sub-system requests a service from another sub-system by passing a message to it. A return message includes the results of executing the service. In this stage, the module-level interfaces and the system-level interfaces were tested thoroughly for errors.

Validation testing

Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can be reasonable expected by the users. After validation test has been conducted, one of two possible conditions exists:

- A derivation from specification is uncovered and a deficiency list is created.
- The function or performance characteristics confirm to specification and are accepted.

Output testing

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specific format. The outputs generated or displayed by the system under consideration are tested by asking the users about the format required by them.

The outputs produced by the newly developed system were shown to the users and suggestions regarding the screen formats were invited. Those suggestions that were found vital were taken into account.

User acceptance testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of development and making changes wherever required.

As this system was developed under constant discussion with the users of the system, the changes requested by them were made during the system development stage itself.

During the above phases of testing Online Monitoring revealed some of the errors were detected and corrected then and there. Testing cannot show the absence of the defect. Hence during the implementation of the commercial plan system undetected error, if any, may be detected and it forms the part of maintenance.

5.2 TEST CASES

Network communication test

Test case 1: Socket test

The tool uses RMI - REMOTE METHOD INVOCATION to establish connection with the sockets. The RMI initially tests and verifies all the socket functions. Only after the socket test, the other functions are tested. The other operations are carried out only if the basic socket functions respond properly.

Expected result: All the socket functions have to return a true value.

Result: All the socket functions returned a true value indicating that all the functions respond properly.

Test case 2: Connection establishment

The user must be capable of establishing connection and interact with the server.

Expected result: User must connect to any specifies server. If connection is not established, appropriate message should be received.

Result: Call established successfully. Response received successfully.

Authentication test

Test case 1: User validation

Any server has a process for authentication. When a user logs on to a server, the user has to provide a user name and a password. This process is for security.

Expected result: The user should provide a correct login name and a password for the server in order to login as administrator.

Result: The administrator login after providing valid user name and password.

Node collection test

Test case 1: Collecting all nodes

When the administrator want to see all the activated nodes by the name of the node and its IP address, this method should be tested to display all the nodes in the net work.

Expected result: this method should list the all activated node in the network.

Result: This method list all available nodes in the network.

Desktop monitoring test

Test case 1: Call establishment

In order to connect to the remote machine, the administrator has to place a call to the remote machine which is active in the LAN. When the remote machine detects a call, the server accepts the administrator's request and establishes connection.

Expected result: Given an IP address or system name, the remote call should be established followed by establishing connection with the remote machine.

Result: Given a correct IP address or system name the call established successfully. Response received successfully.

Authentication test

Test case 1: User validation

Any server has a process for authentication. When a user logs on to a server, the user has to provide a user name and a password. This process is for security.

Expected result: The user should provide a correct login name and a password for the server in order to login as administrator.

Result: The administrator login after providing valid user name and password.

Node collection test

Test case 1: Collecting all nodes

When the administrator want to see all the activated nodes by the name of the node and its IP address, this method should be tested to display all the nodes in the net work.

Expected result: this method should list the all activated node in the network.

Result: This method list all available nodes in the network.

Test case 2: Response from the active system

Administrator places a call. A response has to be received from the active machine. The active machine sends the call accepted or rejected acknowledgement.

Expected result: Call accepted response should be received.

Result: Response received successfully and acknowledged successfully.

Desktop access test

Test case 1: View only desktop

The administrator can view only the remote desktop. The entire desktop is visible to the administrator.

Expected result: The control of the remote system is not given to the administrator. The administrator is allowed to view active applications on the remote machine. Anything clicked at the user site is not activated at the remote machine.

Result: The remote desktop screen captured successfully and displayed successfully.

Mail server test

Test case 1: ODBC Connection test

The mail server has a database to store mails. This DB must connect with mail server by ODBC.

Expected result: The Database must connect with mail server by ODBC.

Result: The Database must connect with mail server by ODBC successfully.

Test case 2: Administration test

The administrator can do the functions these are working properly.

- Creating a user- New user should be added in the database.
- Deleting a user- The deleted user should be removed from the database.
- Changing user password- The data should be modified in the database.
- View the user list- List the all user in the list box.
- View the message user list-List the user who have messages.
- To compose mail-Compose new mail message.
- To see inbox-List the in coming mails.
- To display help messages-Display help message in the text area.

Expected result: The above functions should be working properly.

Result: The above functions are working successfully.

Mail client test

Test case 1: ODBC Connection test

The mail client has a database to store mails. This DB must connect with mail server by ODBC.

Expected result: The Database must connect with mail client by ODBC.

Result: The Database must connect with mail client by ODBC successfully.

Test case 2: Client functions test

The client can do the such functions these are working properly.

- SignIn –To signin to the application.
- InBox-To see the in coming mails.
- Compose mail-To compose new mail.

Expected result: The above functions should be working properly.

Result: The above functions are working successfully.

5.3. System Implementation

A crucial phase in the systems life cycle is the successful implementation of the new system design. Implementation is the stage of project when the theoretical design is turned into a working system. Implementation involves creating computer-compatible files, training the operating staff, and installing hardware, terminals and telecommunications network before the system is up and running. A critical factor in conversion is not disrupting the functioning of the organization.

In system implementation, user training is crucial for minimizing resistance to change and giving the new system a chance to prove its worth. The training aids includes user manuals, help screens, data dictionary, job aids etc. There are three types of implementation:

- Implementation of a computer system to replace a manual system.
- Implementation of a new computer system to replace an existing one.
- Implementation of a modified application to replace an existing one, using the same computer.

The newly developed Online Monitoring has to be implemented successfully. Training aids for staffs have been provided in the form of help screens. Since organizations system undergoes continual change, the application will undoubtedly have to be maintained. Modifications and changes will be made to the user requirement in order to keep pace with the changing environment.

Software development is incomplete without any documentation. Documentation for the newly developed system is provided to satisfy the following needs.

- Protect the system when personnel are promoted, transferred or leave.
- Represents long-term money savings because it reduced the cost of training.
- Eases system maintenance by centralizing materials describing the system
- Provides a permanence reference on the system.

Documentation of the Online Monitoring has been done at various levels as described below:

Management documentation

Management documentation has the least details but includes

- An overview of the system
- System goals and objective.
- Implementation Details.

Program documentation

The body of every program contains comments explaining the purpose of each module. The comments are included in procedural sections, functions also where complex logic is involved. They may assist maintenance programmers because they reduce the need to read the program line by line when making enhancements.

Refinements based on feedback

The feedback from the operators of the new system will be taken into account and analyzed with great attention and remedial measures will be taken as necessary. Solution will be found for valid & feasible suggestions. The reasons for discarding certain invalid or not-feasible suggestions, if any, will be presented to the user up to their satisfaction

CONCLUSION

6. CONCLUSION

Online Monitoring is successfully designed, developed and tested at the Sovereign InfoTech India Pvt. Ltd., Coimbatore. This project is developed using JAVA_RMI and MSACCESS. This project was done keeping in mind the fact it should follow all the steps of software engineering process and covers the complete Software Development Life Cycle. The users of the software are provided with easy and friendly interface. The user interface provided by this project will be widely accepted by the users in general the software was tested thoroughly to ensure that it works effectively and efficiently. A complete documentation that is provided makes the changes and enhancements that are to be done very easy and provides the vitality of documentation.

The application is tested with the user requirements and verified for validity. The software requirements have been met. Needed documents were generated and adequate documentation has been provided for maintenance and future enhancements

SCOPE FOR FUTURE DEVELOPMENT

7. SCOPE FOR FUTURE ENHANCEMENT

This project for the time being limited only to capture the remote desktop and mailing in the LAN environment. But this can be still developed to

- Monitor many remote systems simultaneously.
- Include voice Chat between the users and the administrator.
- Implement wireless application in an intranet system.

BIBLIOGRAPHY

8. BIBLIOGRAPHY

BOOKS

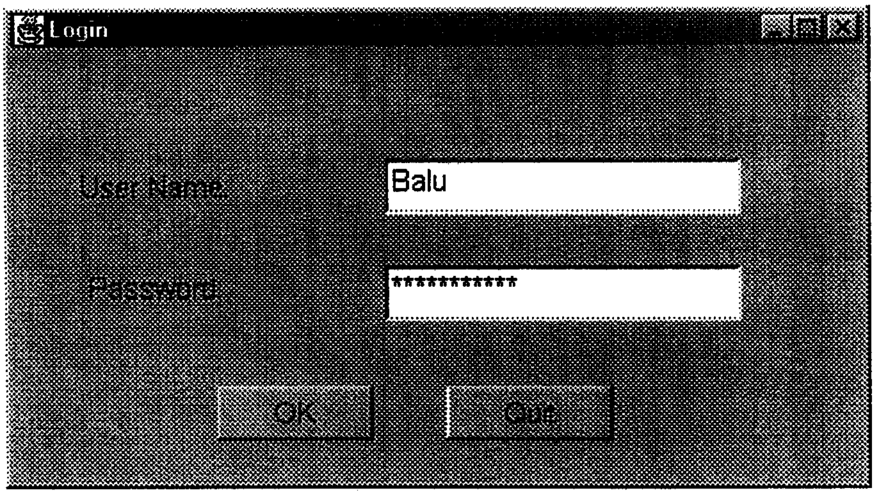
- Troy Bryan Downing "Java RMI: Remote method invocation", IDG Books world wide, INC : Fostercity, California, USA, 2000.
- Herbert Schildt, "Java 2: The complete reference", Tata Mc Graw – Hill publishing company limited, fifth edition, 2002.
- Grady Booch, "Object oriented analysis and Design with applications" person education Asia, Second edition, 2001.
- Ali Bahrami, "Object oriented system development" Mc Graw – Hill International, Edition, 1999.
- Roger S.Pressman, "Software Engineering- A practitioner's approach "Mc Graw – Hill International, Fifth Edition, 2002.

APPENDIX

9. Appendices

9.1. Sample Screens

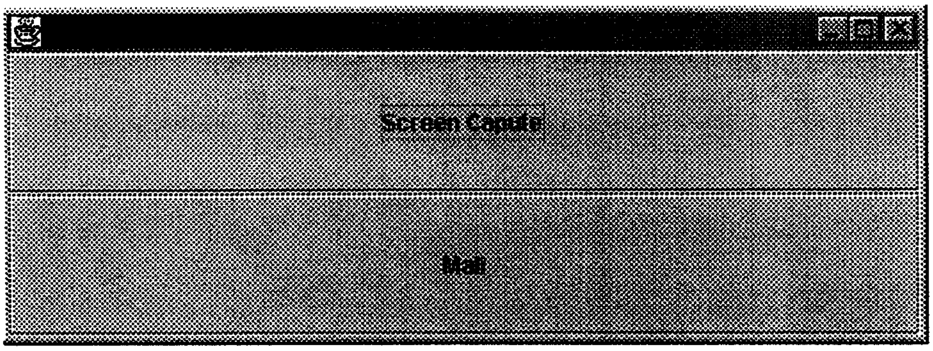
Screen 1 : Authentication.



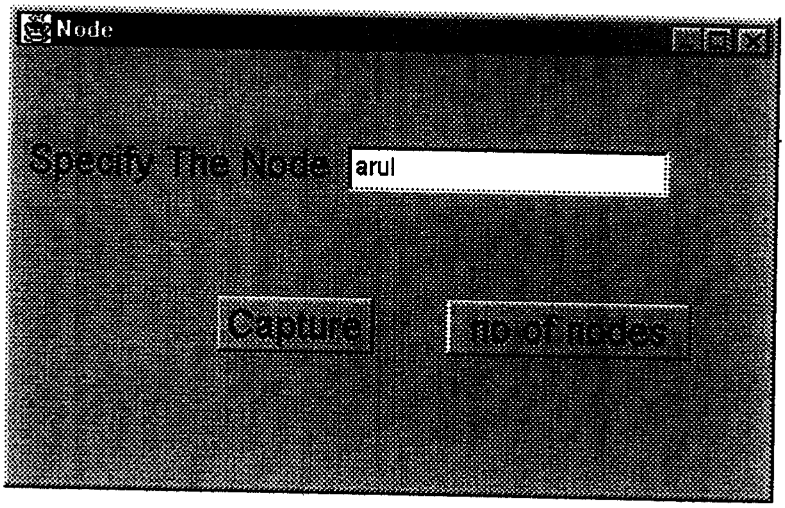
Balu

****

Screen 2 : Application Selection.

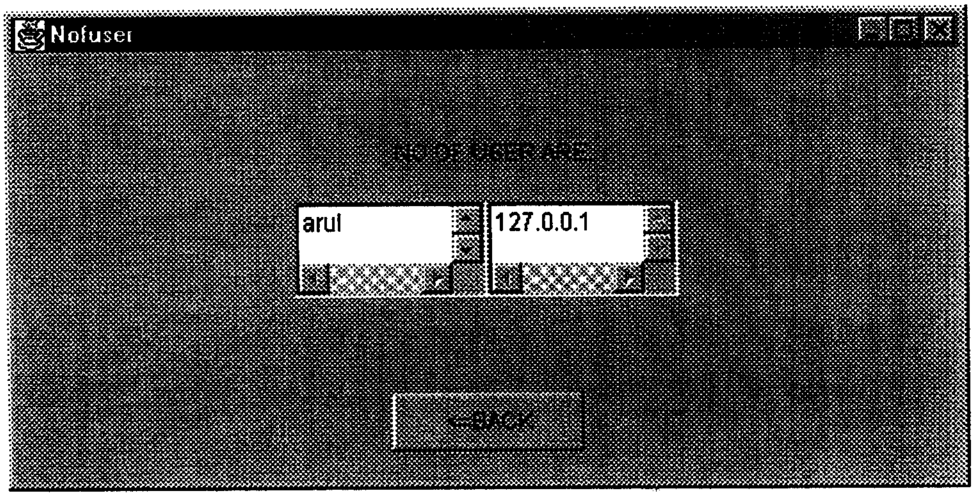


Screen 3 : Node specification screen.



Specify The Node: arul

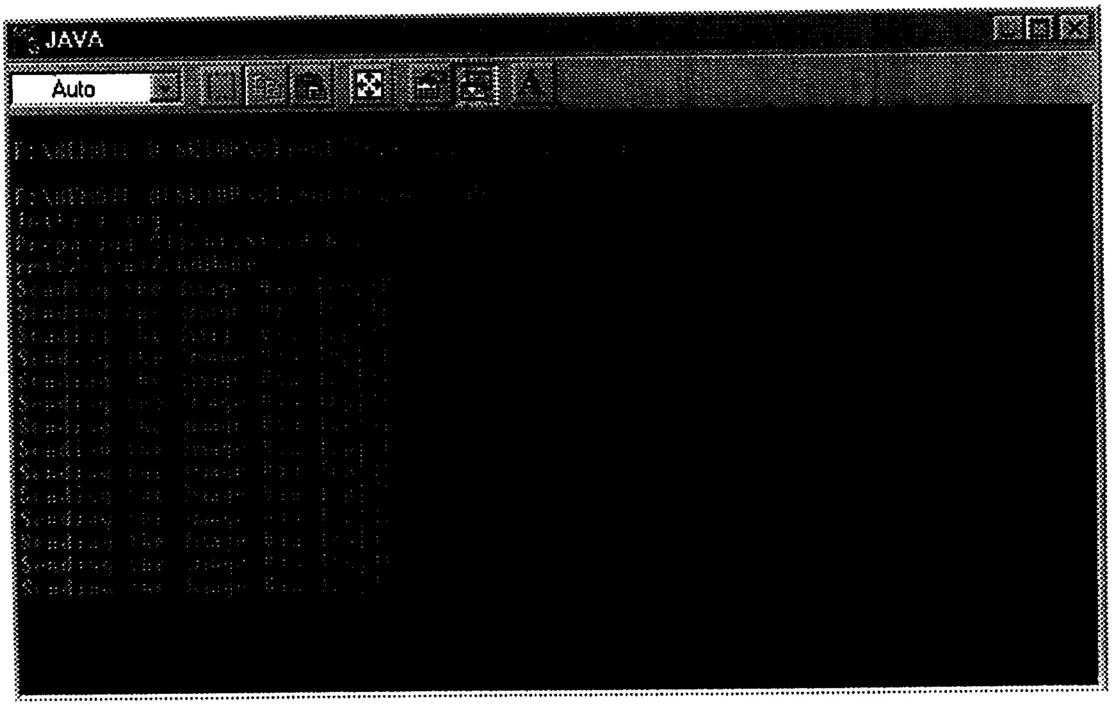
Screen 4 : Available client user screen.



arul

127.0.0.1

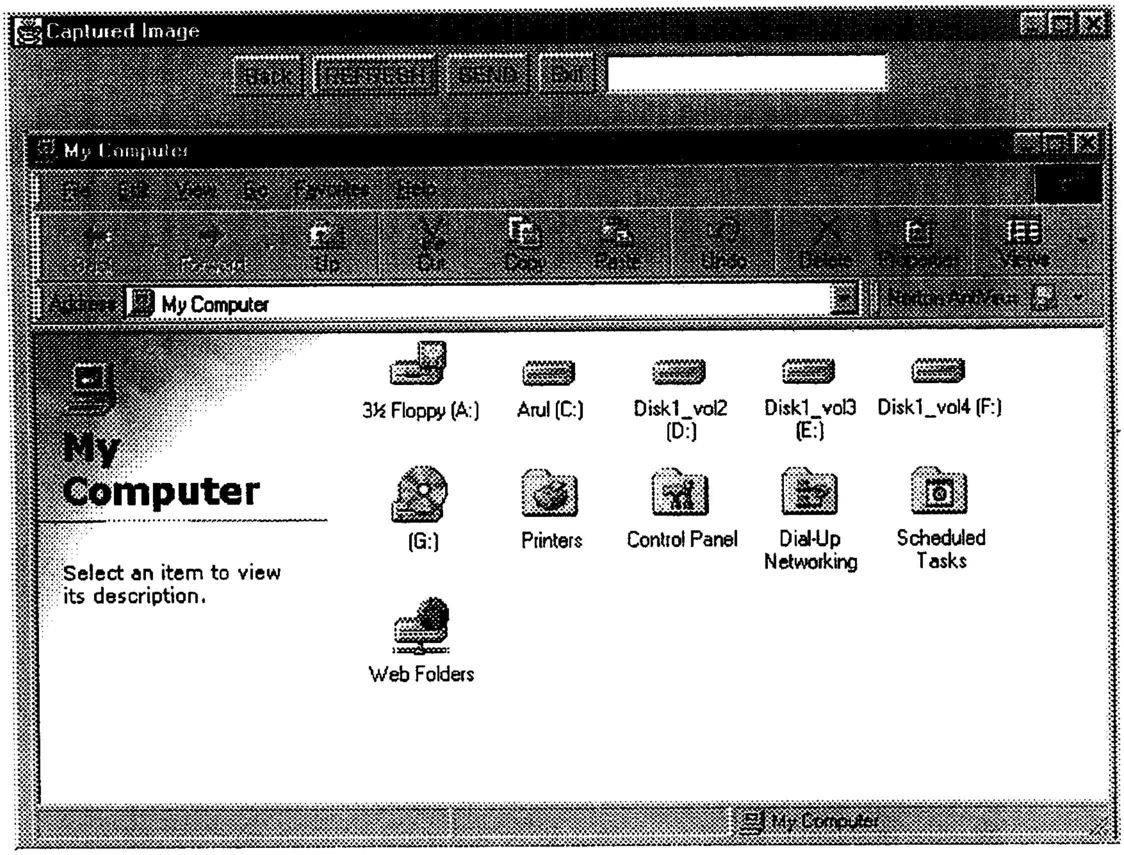
Screen 5 : Image sending serene.



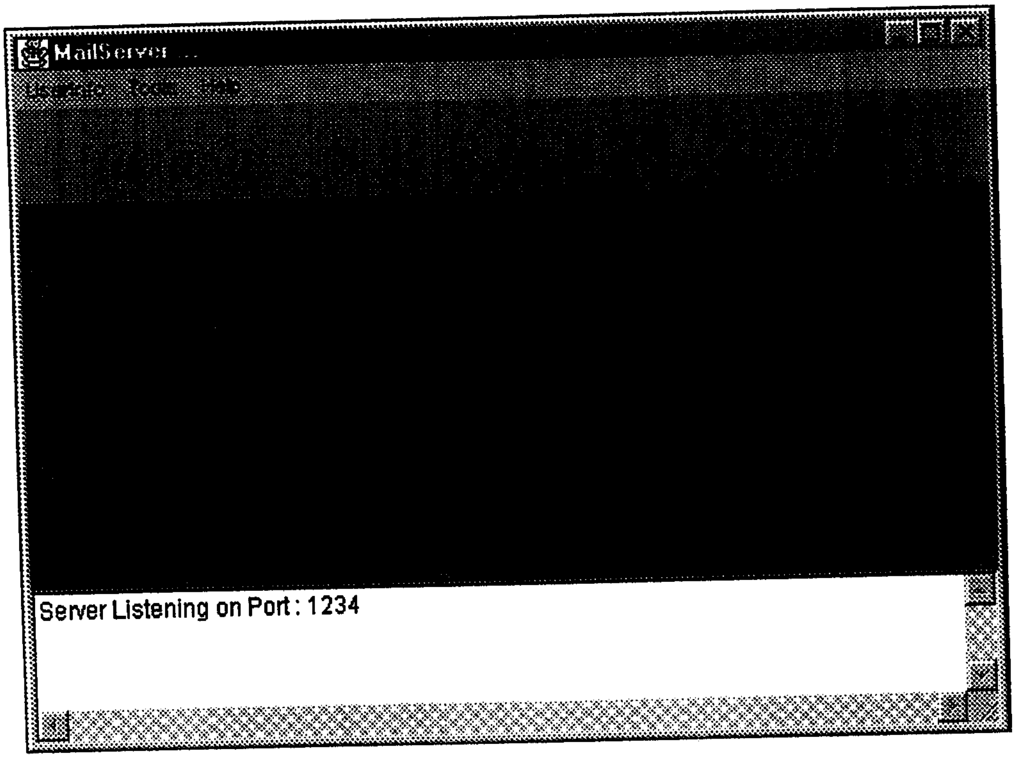
```
[illegible]

[illegible]
[illegible]
[illegible]
[illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
Sending the image [illegible]
```

Screen 6 : Captured image display screen.

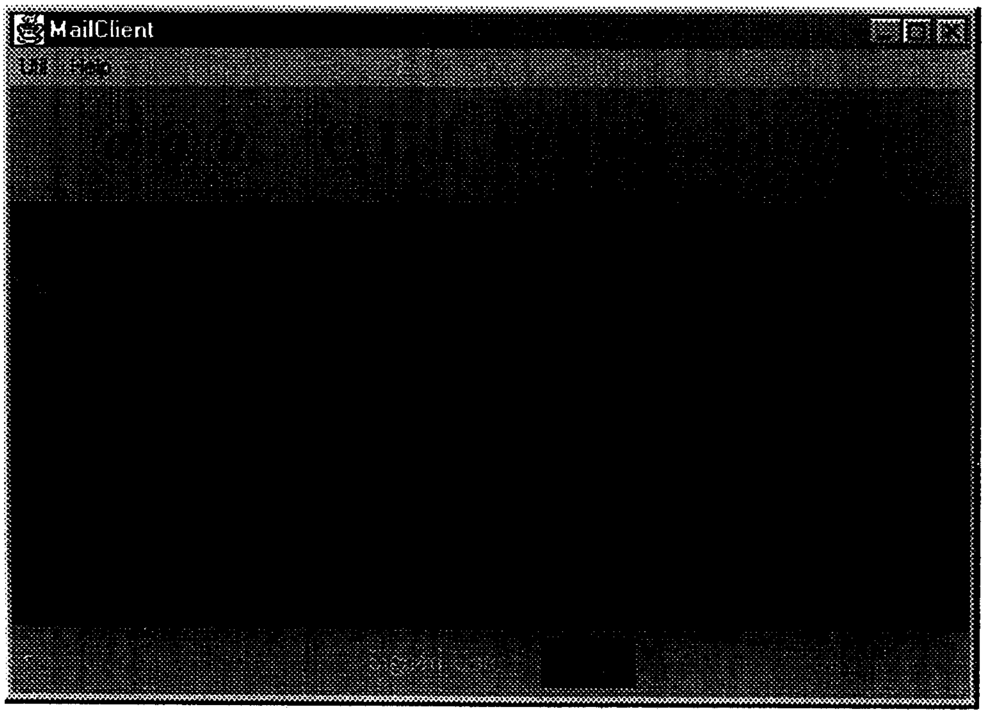


Screen 7 : MailServer Main screen.

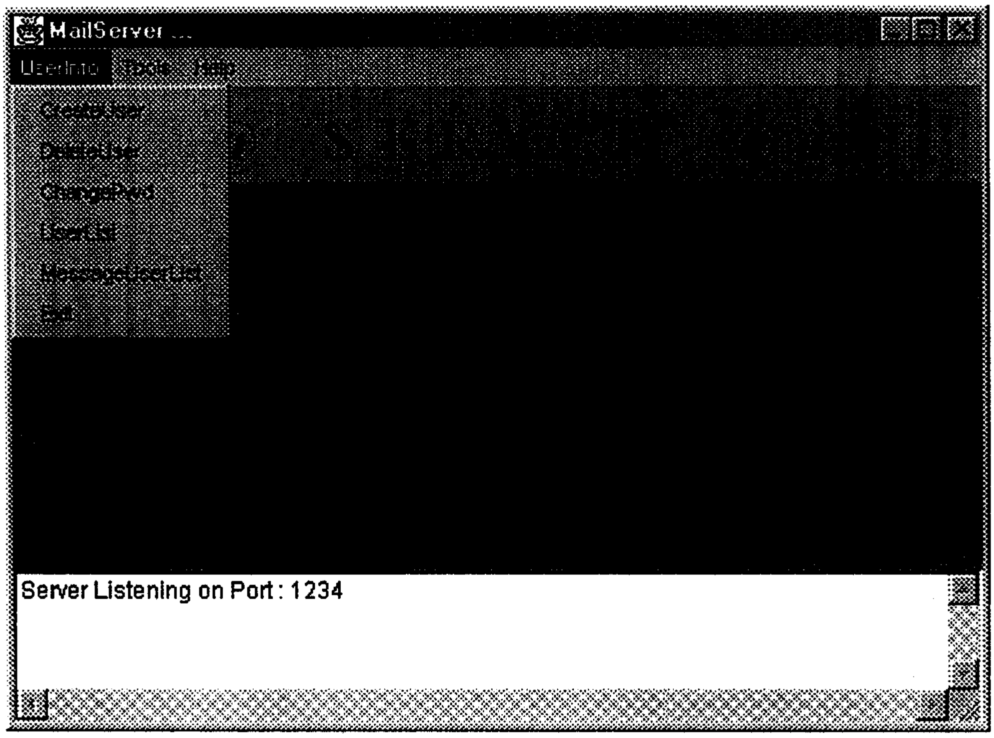


```
Server Listening on Port : 1234
```

Screen 8 : MailClient Main screen.

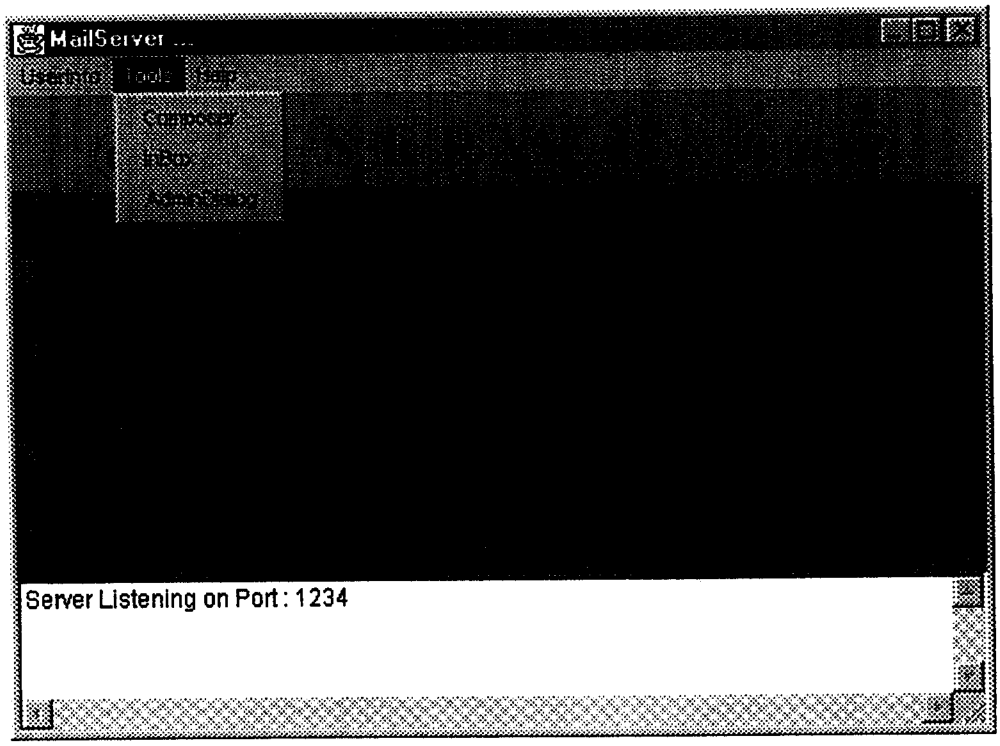


Screen 9 : MailServer's UserInfo menu screen



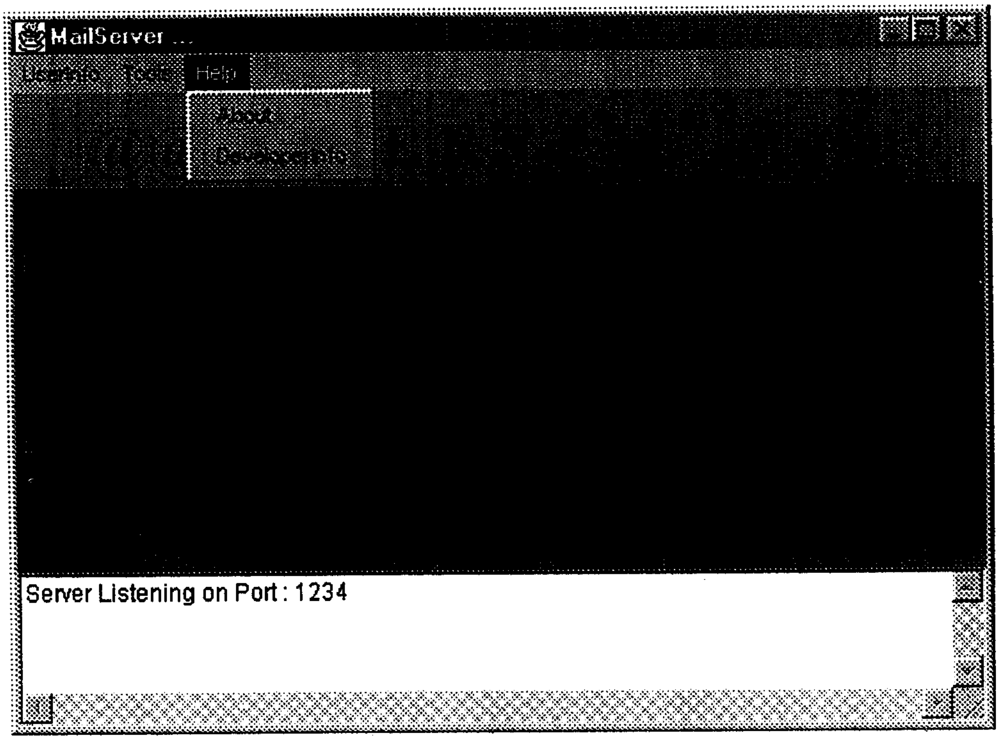
```
Server Listening on Port : 1234
```

Screen 10 : MailServer's Tools menu screen



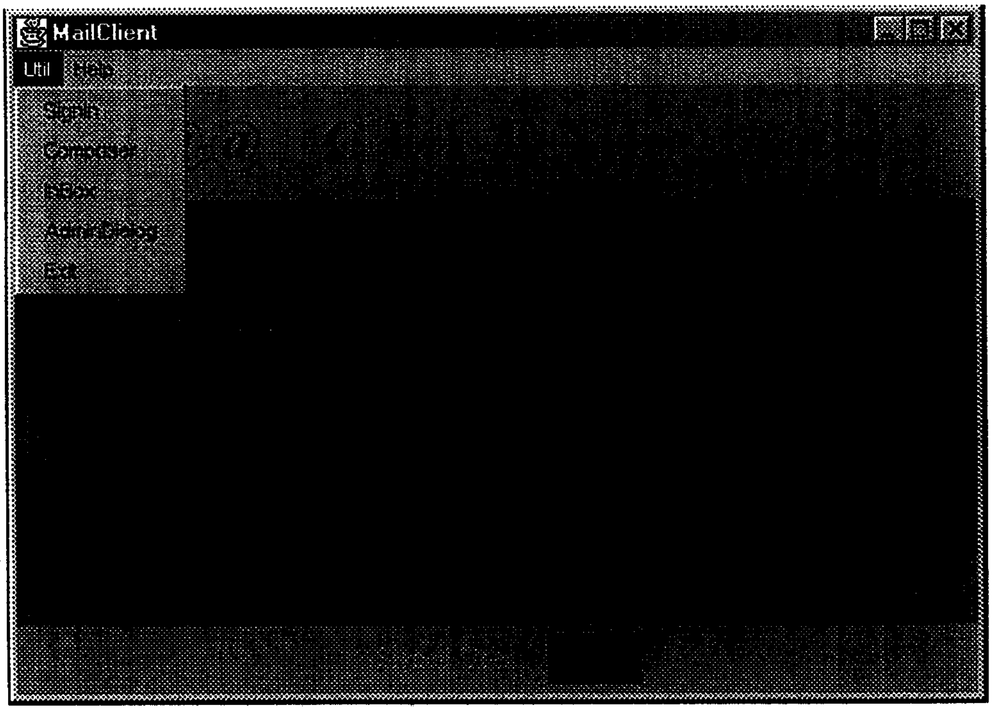
Server Listening on Port : 1234

Screen 11 : MailServer's Help menu screen

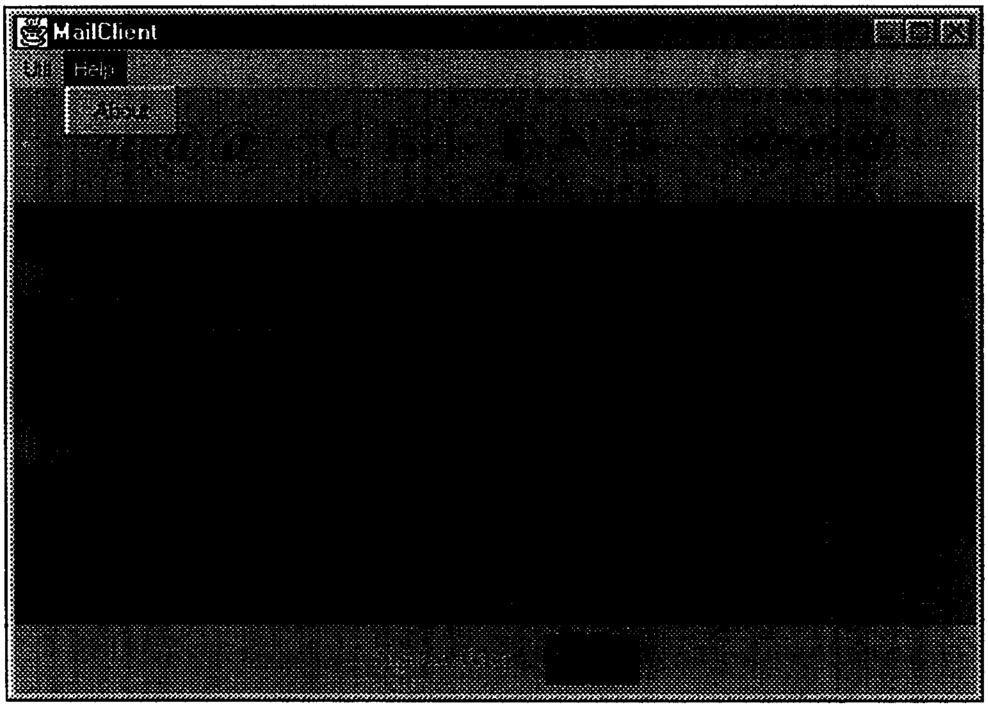


```
Server Listening on Port : 1234
```

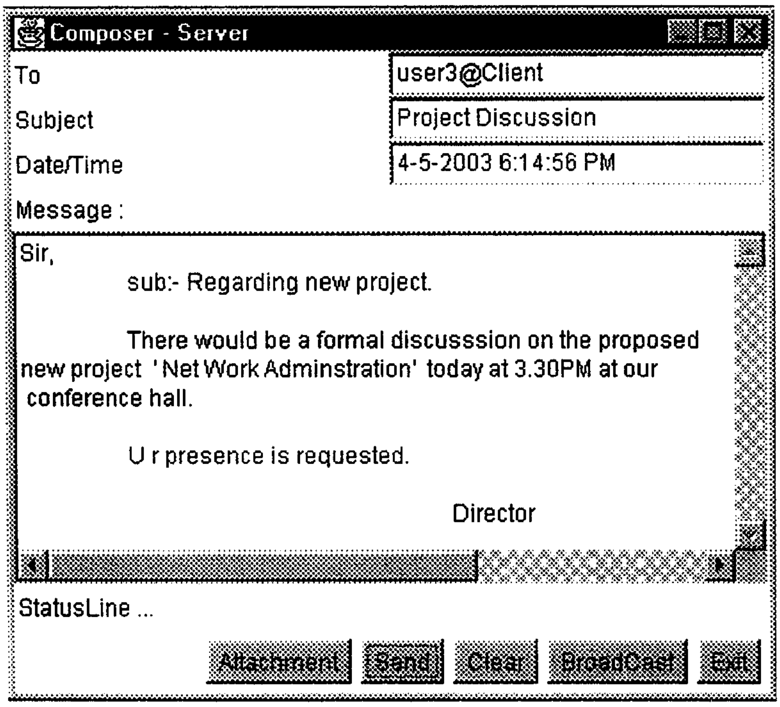
Screen 12 : MailClient's Util menu screen



Screen 13 : MailClient's Help menu screen



Screen 14 : Mail composer screen.



To: user3@Client

Subject: Project Discussion

Date/Time: 4-5-2003 6:14:56 PM

Message :

Sir,

sub:- Regarding new project.

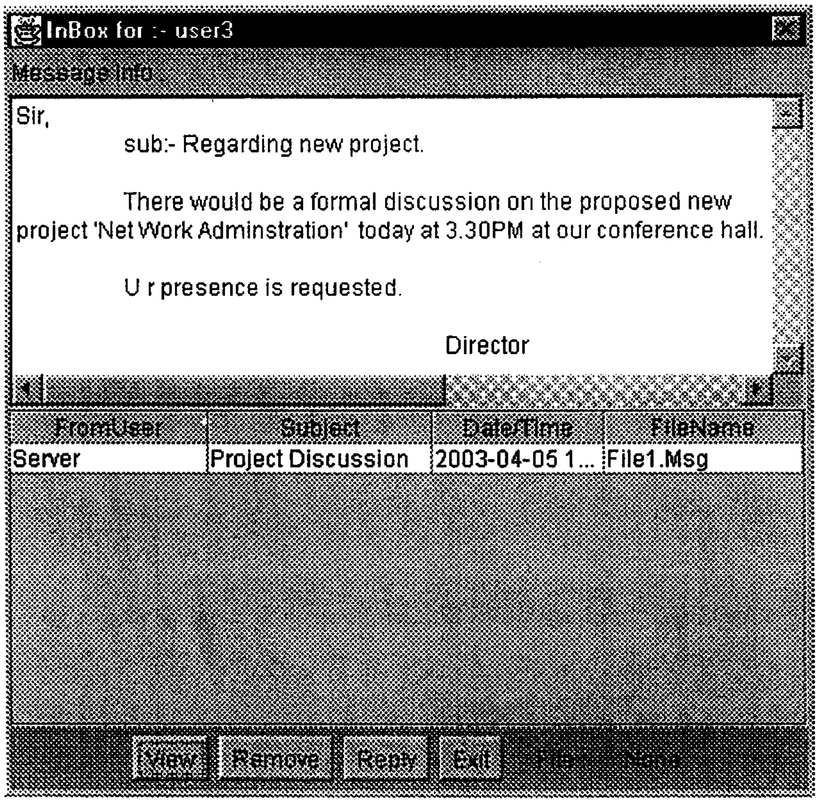
There would be a formal discusssion on the proposed new project 'Net Work Adminstration' today at 3.30PM at our conference hall.

U r presence is requested.

Director

StatusLine ...

Screen 15 : InBox screen.



Sir,
sub:- Regarding new project.

There would be a formal discussion on the proposed new project 'Net Work Adminstration' today at 3.30PM at our conference hall.

U r presence is requested.

Director

FromUser	Subject	Date/Time	FileName
Server	Project Discussion	2003-04-05 1...	File1.Msg

9.2 Sample Coding

```
//User Frame

public class UserFrame extends Frame
{
        //Code for Authentication
}


//End node

public interface EndNode extends Remote
{

        // Interface
}



//End node Implementation


public class EndNodeImpl extends UnicastRemoteObject implements EndNode
{

        public EndNodeImpl() throws RemoteException
        {

                //constructor
        }


        public int[] GetCapturedImagePixels() throws RemoteException
        {
                //Get Image Pixels
        }

        public int GetScreenWidth() throws RemoteException
        {
                //Get Screen Width
        }
        public int GetScreenHeight() throws RemoteException
        {
                // Get Screen Height
        }



//Selector


public class Selector extends Frame
{
        public Selector()
        {
                //Constructor
        }

}


//no of users

class Nofuser extends Frame
{

         Nofuser()
        {
                // Display all online users
        }

}



//Screen capture


public class CaptureScreen extends Frame
{

        public CaptureScreen()
        {
                //Capturing Screen
        }
}
```

```
//Mail client

class MailClient extends Frame implements ActionListener
{


    MailClient() throws Exception
    {

      // Constructor
    }

    class SignIn extends Dialog implements ActionListener
      {

             SignIn()
             {

      // Signin to the application
             }
      }


}


class Composer extends Frame implements ActionListener
{

      public void Send_Message() throws Exception
      {

// Send message to the administrator
      }


      public void Attachment() throws Exception
      {

             // Send attachments with the message
      }


}
```

```
class InBox extends Dialog implements ActionListener
{

      public void Receive_Rows_fromServer() throws Exception
      {

         // Receive incoming mail from the database
      }


      public void Remove() throws Exception
      {

         // Remove the deleted mail from the database
      }


      public void View () throws Exception
      {

         // View the selected mail message
      }


      public void Reply () throws Exception
      {

         // Send reply  message
      }

}
```

```
//Mail server

class MailServer extends Frame implements Runnable,ActionListener
{


      MailServer() throws Exception
      {

                  // Constructor

      }


      public void Insert_Row_MailInfo() throws Exception
      {

                  // Store the mail in the database
      }



      public void Write_MailtoFile() throws Exception
      {

                  // Store the message contents in a file
      }


 }
```

```
class UserCreation extends Dialog implements ActionListener
{

      UserCreation()
      {

                  // Constructor
      }
      public void Create_User() throws Exception
      {

                  // Creating new user
      }
}

class DeleteUser extends Dialog implements ActionListener
{

      DeleteUser()
      {

         // Constructor
         }


      public void Delete_User() throws Exception
      {

                  // Delete the existing user
      }

}


class UserList extends Dialog implements ActionListener
{

      UserList()
      {

                  // List the all user
      }
}
```

```
class ChangePassWd extends Dialog implements ActionListener
{

    public void Change_PassWord() throws Exception
    {

            // Change the user password
    }
}

class MessageUserList extends Dialog implements ActionListener
{

    MessageUserList()
    {

            // List the users who received message

    }
}




class HelpDlg extends Dialog
{
    HelpDlg()
    {

            // Display help message
    }
}

class ServComposer extends Frame implements ActionListener
{
    ServComposer()
    {

            // Constructor
    }
```

```
        public void Send() throws Exception
        {

                // Send message to the user
        }

        public void BroadCast() throws Exception
        {

                // Send message to all user
        }



          public void Attachment() throws Exception
            {

                // Send attachments with the message
            }

    }

    class ServInBox extends Frame implements ActionListener
    {


        ServInBox()
        {

        }
        public void Receive_Rows() throws Exception
        {


         // Receive incoming mail from the database
        }



    public void Remove() throws Exception
        {

           //Remove the deleted mail from the database
        }
```

```
          public void View () throws Exception
          {

             // View the selected mail message

          }


      public void Reply() throws Exception
          {

             // Send reply  message

          }


      }
```