



REMOTE- ADMIN SERVICE CONTROL

A PROJECT REPORT

Submitted by

KRITHIGA.M 71201205025

LAKSHMI NAIR 71201205026



in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2005

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**REMOTE-ADMIN SERVICE CONTROL**” is the bonafide work of “**Ms. KRITHIGA.M and Ms. LAKSHMIN NAIR**” who carried out the project work under my supervision.



SIGNATURE

Dr.S.Thangasamy

HEAD OF THE DEPARMENT

Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore – 641006



SIGNATURE

Mr.K.R.Baskaran

SUPERVISOR

Assistant Professor

Dept. of INFORMATION TECHNOLOGY

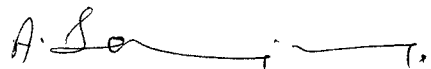
Kumaraguru College of Technology

Coimbatore – 641006

The candidates with University Register Nos. 71201205025, 71201205026 were examined by us in the project viva-voce examination held on 19.04.2005.



INTERNAL EXAMINER



EXTERNAL EXAMINER

19/05/04

ABSTRACT

The Remote-Admin Service Control System has been designed for anyone who wishes to have control of their OS anywhere by using a GPRS enabled Mobile Phone. In order to control the OS, we make use of an **intermediate server** to transmit the commands between the Mobile and the PC to perform the necessary operations. The OS services and processes can be controlled through the Mobile.

Multiple users can use the intermediate server (one OS can be controlled by only one mobile at a time). Initially, both the mobile user and the PC client is registered with the intermediate server. The intermediate server accepts the request from the mobile and recognizes the command and sends it to the PC Client where it is executed.

Finally, the Remote Admin Service Control System provides a user friendly interface for remote users. It also focuses on security aspects by allowing only authorized mobile users to access the PC.

ACKNOWLEDGEMENT

We wish to express my sincere thanks to **Dr.K.K.Padmanaban** Ph.D., Principal, Kumaraguru College of Technology, Coimbatore for permitting us to undertake this project.

We express our deep sense of gratitude to **Dr.S.Thangasamy** Ph.D., Head Of the Department, Computer Science and Engineering, for his valuable guidance, assistance and encouragement for the successful completion of this project.

We also extend our heartfelt thanks to our course Co-coordinator and our guide **Mr.K.R.Baskaran**, Assistant Prof., Department of Information Technology for spending his time with us and helping us out during the phases of this project.

We thank all the teaching and non-teaching staff of our department for providing us the technical support in the duration of our project.

We also thank all our friends who helped us to complete our project successfully.

DECLARATION

We

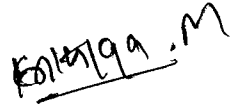
Krithiga.M 71201205025

Lakshmi Nair 71201205026

Declare that the project entitled “**REMOTE ADMIN SERVICE CONTROL**”, submitted in partial fulfillment to Anna University as the project work of Bachelor Of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of **Mr.K.R.Baskaran**, Assistant Prof., Department of Information Technology , Kumaraguru College Of Technology, Coimbatore.

Place: Coimbatore

Date :

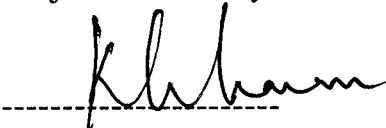


[**Krithiga.M**]



[**Lakshmi Nair**]

Project Guided by



[**Mr.K.R.Baskaran.**]

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE
	ABSTRACT	V
	LIST OF TABLES	VIII
	LIST OF FIGURES	VIII
1.	INTRODUCTION	1
	1.1 Description	1
	1.2 Problem Description	1
	1.3 Objectives of Project	2
2.	SYSTEM ANALYSIS	3
	2.1 Feasibility Study	3
	2.1.1 Existing System and Limitations	3
	2.1.2 Proposed System and Advantages	4
	2.2 Hardware Requirements	5
	2.3 Software Requirements	5
	2.4 Software Overview	6
	2.4.1 Java 2 Micro Edition	6
	2.4.2 Visual Basic	9
	2.4.3 Introduction to Java	12
3.	SYSTEM DESIGN	16
	3.1 Fundamental Design Concepts	16
	3.2 Database Design	16
	3.2.1 Table Design	17
	3.3 Overall Architecture	18

CHAPTER NO.	TITLE	PAGE
	3.4 UML Diagrams	19
	3.4.1 Use-Case Diagram	19
	3.4.2 Class Diagram	20
	3.4.3 Interaction Diagram	21
	3.4.4 Collaboration Diagram	22
4.	SOFTWARE IMPLEMENTATION MODEL	23
	4.1 Module Description	23
	4.1.1 Mobile Server Connection	23
	4.1.2 Client User Interface Design	23
	4.1.3 Intermediate Server Connection	24
	4.1.4 Invoking Admin Services	24
	4.2 Input/Output Design Formats	28
	4.2.1 Client Side Logging in	28
	4.2.2 Mobile User End	28
5.	PRODUCT TESTING	29
	5.1 Testing Methods	29
	5.1.1 Unit Testing	29
	5.1.2 Validation Testing	30
	5.1.3 Output Testing	30
	5.1.4 Integration Testing	30
6.	FUTURE ENHANCEMENTS	31
7.	CONCLUSION	32
8.	APPENDIX	33
9.	REFERNCES	63

LIST OF TABLES

<u>TABLE NO</u>	<u>TABLE NAME</u>	<u>PAGE NO</u>
3.2.1	User_access	17

LIST OF FIGURES

<u>FIGURE NO</u>	<u>FIGURE DESCRIPTION</u>	<u>PAGE NO</u>
3.3.1	Overall Architecture	18
3.3.2	Use Case Diagram	19
3.3.3	Class Diagram	20
3.3.4	Interaction Diagram	21
3.3.5	Collaboration Diagram	22
8.1	Main Screen	58
8.2	Login Screen	59
8.3	Registration Form	60
8.4	Mobile User Screen	61
8.5	Display of Services	62

1. INTRODUCTION

1.1 DESCRIPTION:

The main aim of the project is the remote control of administrator services in the PC. The command for the various services to be controlled are sent or selected through the remote mobile user, then passed on to the intermediate server and finally invoked in the client side user. The project involves the controlling of a single PC by the corresponding registered mobile user. Both the mobile and the PC should be registered in order to control the services.

1.2 PROBLEM DESCRIPTION:

This Project tries to envisage the ability to control the administrator services of the PC when its user is not physically present in front of the system. The Client initially registers itself to the system using the registration form .An already registered user gets connected to the intermediate server the moment he signs in.

At the other end, the mobile user enters his username with which he has already registered and the corresponding password. Both the user name and the password go to the server for verification where it is checked with the database that stores the username and password of every registered user.

The admin service that the user wishes to control is selected from the mobile. Each command has a particular header attached to it .When the selection is made the header is passed to the server and finally given to the client side .At the client side the corresponding service is invoked when the header is recognized .The same is followed when a service is to be stopped .

In addition to the admin services that are to be invoked the “File open” application is also invoked when the header is recognized at the client side , shell command is used to open the specified file for the user at the PC.

The user operating the PC currently can also be recognized and information is sent to the mobile user if asked for.

Other operations such as shut down and log off are also invoked in case of emergency from a remote location.

1.3 OBJECTIVES OF THE PROJECT

The main objective of the project is as listed:

- To control the admin services of the PC from a remote location
- To open a file for the user at the client side
- To log off and shut down the system in case of emergency
- To get information about the user accessing the PC at any instant.

2. SYSTEM ANALYSIS

This is the phase where all the requirements needed for developing and implementing the project is gathered and placed as documentation for further reference. This serves as a manual for the users for easy use of the system. For implementing this project the needed requirements are:

2.1 FEASIBILITY STUDY

The main objective of the feasibility study is to test the technical feasibility of developing a computer system. The assessment of technical feasibility is based on the outline requirements of the system requirements in terms of inputs, outputs, files, programs and procedures. The system is built with J2ME technology (mobile end), java (intermediate server) and Visual Basic (client side).

2.1.1 EXISTING SYSTEM AND ITS LIMITATIONS:

As of now, facilities are available to access any information from the internet remotely without the help of a PC through the mobile using the GPRS connection but no such system exists that can control the administrator services with the help of a mobile through an intermediate server. The system that has been designed in this project enables the control of these services even when the user is not present around the system.

LIMITATIONS:

- Requires the administrator to be physically present near the PC to control the services

2.1.2 PROPOSED SYSTEM AND ITS ADVANTAGES:

The proposed system focuses on the shortcomings of the conventional network monitoring and handling kits. In addition to carrying on the basic functionalities of the conventional systems 'Mobile Admin' aims to bring about a new system that is more flexible, user friendly and allows the administrator to remain mobile while having total control over the network. This process is made effective by working on the drawbacks of the previous system.

ADVANTAGES :

- Allows the mobile user to control his network from anywhere in the world as long as he has a GPRS connection.
- Includes a wide array of functions to perform commonly used functions with an extremely user friendly interface which requires no training

2.2 HARDWARE REQUIREMENTS:

Processor	:	Pentium IV
Speed	:	2.2GHz
RAM size	:	128 MB RAM
Hard disk capacity	:	40GB

2.3 SOFTWARE REQUIREMENTS:

2.3.1 Mobile User End:

Operating System	:	Windows XP
Database Used	:	Microsoft Access
Language	:	J2ME Wireless Toolkit

2.3.2 Intermediate Server

Operating System	:	Windows XP
Database Used	:	Microsoft Access
Language	:	JDK 1.4, Java

2.3.3 PC Client End

Operating System	:	Windows XP
Database Used	:	Microsoft Access
Language	:	Visual Basic 6.0

2.4 SOFTWARE OVERVIEW

2.4.1 Java 2 Micro Edition (J2ME)

The J2ME Wireless Toolkit supports the development of Java applications that run on devices compliant with the Mobile Information Device Profile (MIDP) version 2.0, such as cellular phones, two-way pagers, and communicators. In addition, the Wireless Toolkit supports development of applications compliant with the Wireless Messaging API (WMA) and the Mobile Media API (MMAPI).

2.4.1.1 Features of the Wireless Toolkit:

The KToolBar, included with the J2ME Wireless Toolkit, is a minimal development environment with a graphical user interface (GUI) for compiling, packaging, and executing MIDP applications. The only tools there is a need for is a third-party editor for Java source files and a debugger.

An IDE compatible with the J2ME Wireless Toolkit, such as the Sun™ One Studio 4, Mobile Editor, provides even more convenience. While using the Wireless Toolkit within an IDE, it is possible to edit, compile, package, and execute or debug MIDP applications, all within the same environment.

2.4.1.2 Compiling, Preverifying, and Debugging:

When compiling MIDlets through the KToolBar (or an IDE compatible with the toolkit), the source files are compiled using the J2SE™ SDK compiler. Preverification of the compiled files is done with the Preverifier

that prepares class and JAR files and class directories. Preverification takes place immediately after compilation. It is possible to debug applications within the environment using the Emulator, which simulates the execution of the application on various devices.

2.4.1.3 Packaging:

It is possible to package your MIDlet suite from the KToolBar or with a compatible IDE. The KToolBar gives the choice of creating a standard package or creating an obfuscated package that produces a smaller JAR file by reducing the size of the classes in the suite through the obfuscation process.

2.4.1.4 Authenticating and Authorizing MIDlets:

Trusted applications can be created that have permission to use protected APIs. Permission can also be requested to access network protocol APIs through the Project Settings dialog box from the KToolBar. The MIDlet suite can be signed and assigned a security domain that defines the suite's authorization level with the Sign MIDlet Suite window.

2.4.1.5 Performance Tuning:

The Wireless Toolkit's Profiler enables us to optimize the performance of the MIDlet suite by determining where bottlenecks might be occurring during runtime. Execution time of the MIDlet suite can be improved by examining the time spent in method calls, the number of times a method is called during runtime, and the amount of time a method runs compared to the overall runtime of the application. The performance speed of the application can

be adjusted in the Performance panel of the Project Settings dialog box. Setting the speed features does not demonstrate how the application would run on an actual device; however, by adjusting the speed emulation parameters, a better representational performance of the application on a device can be achieved.

2.4.1.6 Memory and Network Monitoring:

The Wireless Toolkit provides tools to examine and analyze memory usage by the application and network transmissions between the device and the network. It is possible to get an overall view of memory usage during runtime of the application and get a breakdown of memory usage per object to see where in the application memory usage can be optimized. With the Network Monitor, network transmissions for several types of network protocols can be examined.

2.4.1.7 Working with the Emulator:

The J2ME Wireless Toolkit comes with a selection of emulated devices to run and test the applications on. Representations of mobile devices are available from the Device list on the KToolBar. The emulated devices are capable of emulating the features in the CLDC, MIDP, MMAPI, and WMA specifications. The functionality for an emulated device through the Preferences Window can be set. Various emulator utilities such as the Profiler, the Network Monitor, the Memory Monitor, and the Certificate Manager from the Utilities window are also present.

2.4.2 Visual Basic:

Visual Basic is a member of the Visual Studio 6.0 family of development products, which includes:

- Visual Basic
- Visual C++
- Visual FoxPro
- Visual InterDev
- Visual J++
- Visual SourceSafe
- MSDN Library

Visual Basic provides a complete set of tools to simplify rapid application development.

The "Visual" part refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, prebuilt objects can be added into place on screen.

The "Basic" part refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language.



The Visual Basic programming language is not unique to Visual Basic. The Visual Basic programming system, Applications Edition included in Microsoft Excel, Microsoft Access, and many other Windows applications uses the same language. The Visual Basic Scripting Edition (VBScript) is a widely used scripting language and a subset of the Visual Basic language.

Data access features allow creation of databases, front-end applications, and scalable server-side components for most popular database formats, including Microsoft SQL Server and other enterprise-level databases.

ActiveX™ technologies allow the use of functionality provided by other applications, such as Microsoft Word processor, Microsoft Excel spreadsheet, and other Windows applications. Even automate applications and objects created using the Professional or Enterprise editions of Visual Basic.

Internet capabilities make it easy to provide access to documents and applications across the Internet or intranet from within the application, or to create Internet server applications.

The finished application is a true .exe file that uses a Visual Basic Virtual Machine

2.4.2.1 Visual Basic Editions:

Visual Basic is available in three versions, each geared to meet a specific set of development requirements.

2.4.2.2 Learning Edition:

The Visual Basic Learning edition allows programmers to easily create powerful applications for Microsoft Windows and Windows NT®. It includes all intrinsic controls, plus grid, tab, and data-bound controls.

2.4.2.3 Professional Edition:

The Professional edition provides computer professionals with a full-featured set of tools for developing solutions for others. It includes all the features of the Learning edition, plus additional ActiveX controls, the Internet Information Server Application Designer, integrated Visual Database Tools and Data Environment, Active Data Objects, and the Dynamic HTML Page Designer.

Enterprise Edition:

The Enterprise edition allows professionals to create robust distributed applications in a team setting. It includes all the features of the Professional edition, plus Back Office tools such as SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, and more.

Winsock Control:

A WinSock control allows the connection to be created to a remote machine and exchange data using either the User Datagram Protocol (UDP) or the Transmission Control Protocol (TCP). Both protocols can be used to create client and server applications. Like the Timer control, the WinSock control doesn't have a visible interface at run time.

Selecting a Protocol:

When using the WinSock control, the first consideration is whether to use the TCP or the UDP protocol. The major difference between the two lies in their connection state:

- The TCP protocol control is a connection-based protocol, and is analogous to a telephone — the user must establish a connection before proceeding.
- The UDP protocol is a connectionless protocol, and the transaction between two computers is like passing a note: a message is sent from one computer to another, but there is no explicit connection between the two. Additionally, the maximum data size of individual sends is determined by the network.

2.4.3 Introduction to JAVA:

Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets. Because of its rich set of API's, similar to Macintosh and Windows, and its platform independence, Java can also be thought of as a platform in itself. Java also has standard libraries for doing mathematics.

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be

downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer

The inventors of Java wanted to design a language, which could offer a solution to some of the problems encountered in modern programming. They wanted the language to be not only reliable, portable and distributed.

Although the above appears to be a list of buzzwords, they aptly describe the full potential of the language. These features have made Java the first application language of the World Wide Web. Java will also become the premier language for general-purpose stand-alone applications.

2.4.3.1 Properties of java:

- Compiled and interpreted
- Platform-Independent and Portable
- Robust and Secure
- Multithreading and Interactive
- Dynamic and Extensible

Compiled and Interpreted:

Usually a computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system. First, Java compiler translates source code into what is known as byte code instructions.

Platform Independent and Portable:

The most significant contribution of Java over other languages is its portability. Java programs can be easily moved from one computer system to another, anywhere anytime. Changes and upgrades in operating systems, processors and system resources will not force any changes in Java programs. This is the reason why Java has become a popular language for programming on Internet, which inter connects different kinds of systems worldwide. We can download a Java applet from a remote computer on to our local system via Internet an extension of the user's basic system providing practically unlimited number of accessible applets and applications.

Robust and Secure:

Security becomes an important issue for a language that is used for programming on Internet. Threat of virus and abuse of resources is everything. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

Distributed:

Java is designed as a distributed language for creating applications on networks. It has the ability to share both data programs. Java applications can open and access remote objects on Internet as easily as they can in a local system. This enables multiple programmers at multiple remote locations to collaborate and work together on a single project.

Multithreaded:

Multithreaded means handling multiple tasks simultaneously. Java supports multithreading programs. This means that we need not wait for the application to finish one task before beginning another. For example, we can listen to an audio clip time download an applet from a distance computer. This feature greatly improves the interactive performance of graphical applications.

Dynamic and Extensible:

Java is a dynamic language. Java is capable of dynamically linking in new class libraries methods and objects. Java can also determine the type of class through a query, making it possible to either dynamically link or abort the program, depending on the response

2.4.3.2 Advantages of JAVA

Java has gained enormous popularity since it first appeared. Its rapid ascension and wide acceptance can be traced to its design and programming features, particularly in its promise that we can write a program once, and run it anywhere. As stated in Java language white paper by Sun Microsystems: "Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, multithreaded, and dynamic."

3. SYSTEM DESIGN

3.1 FUNDAMENTAL DESIGN CONCEPTS:

A Software design is a model of a real world system that has many participating entities and relationships. This design is used in a number of different ways. It acts as a basis for detailed implementation. It serves as a communication medium between the designers of the subsystems; it provides information to system maintainers about the original intentions of the system designers.

System design is a transition from user oriented documents to document oriented programmers or database personnel. Design phase acts as a bridge between the software requirement specification and implementation phase, which satisfies the requirements.

3.2 DATABASE DESIGN

A Database is a repository for stored operational data. It is simply a collection of interrelated data stored with minimum redundancy to serve many users quickly and effectively. The main objective in the database concept is to bring data integration so as to allow several users to share a common data for various applications. It facilitates quick and easy access for all the users .The maintenance of database is done by the DBMS.

The major objectives for maintaining such databases are:

- Reduced redundancy
- Enforcing Standards
- Sharing of Data

- Maintaining Integrity
- Data Independence
- Applying Security Restrictions

3.2.1 TABLE DESIGN

FIELD NAME	TYPE	WIDTH	DESCRIPTION
S.NO	NUMBER	3	
M_UNAME	VARCHAR	50	Mobile User Name
M_PASSWORD	VARCHAR	50	Mobile Password
C_UNAME	VARCHAR	50	Client User Name
C_PASSWORD	VARCHAR	50	Client Password

Table 3.2.1: User_access

3.3 OVERALL ARCHITECTURE

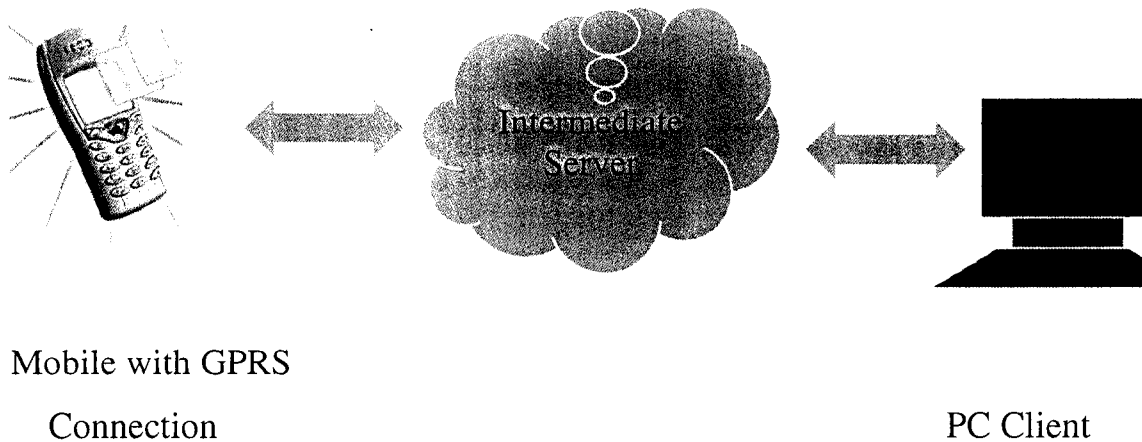


Figure 3.3.1 Overall Architecture

The architecture emphasizes on the fact that the services are controlled when the commands are passed from the mobile user to the client side through the intermediate server.

Multiple users can use the intermediate server (one OS can be controlled by only one mobile at a time)

3.4 UML DIAGRAMS

3.4.1 USE CASE DIAGRAM

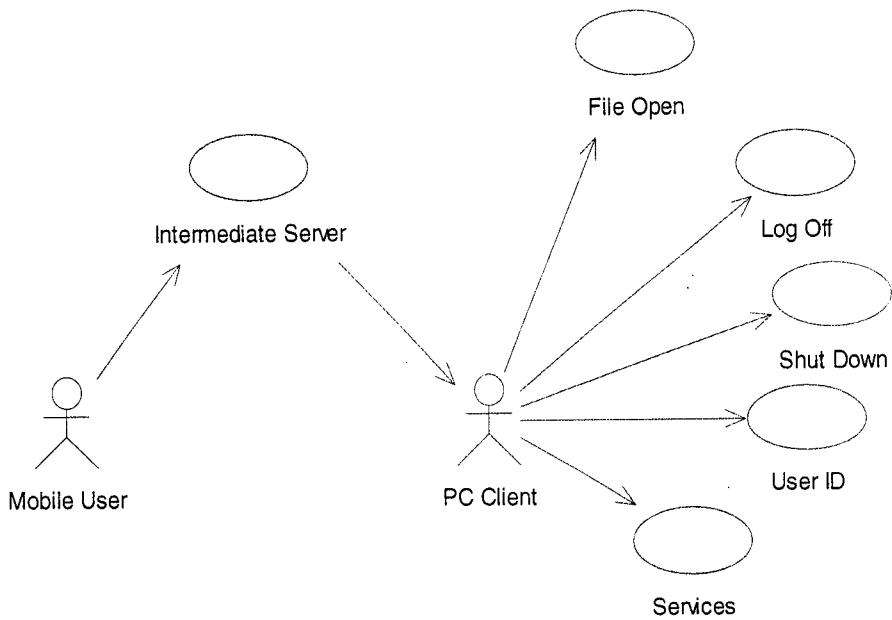


Figure 3.4.1 Use Case Diagram

3.4.2 CLASS DIAGRAM

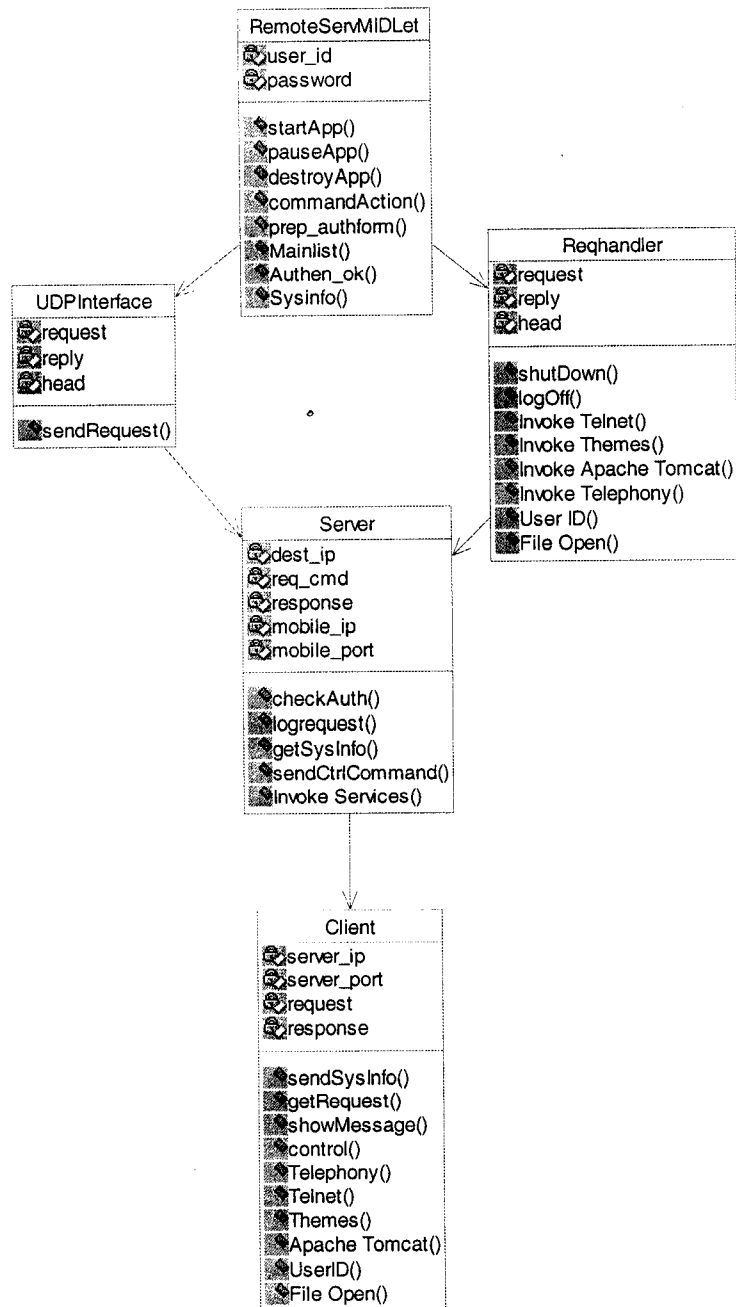


Figure 3.4.2 Class Diagram

3.4.3 INTERACTION DIAGRAM

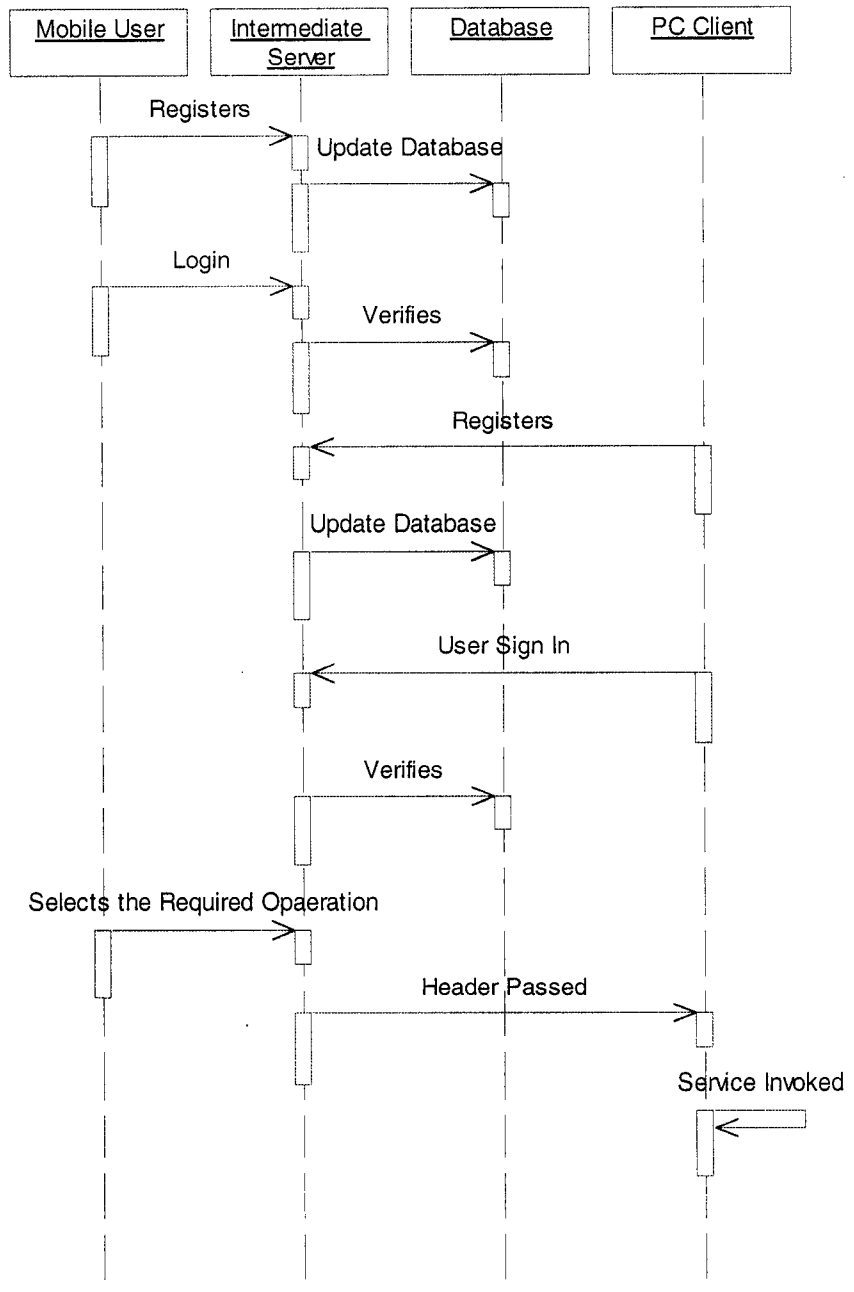


Figure 3.4.3 Interaction Diagram

3.4.4 COLLABORATION DIAGRAM

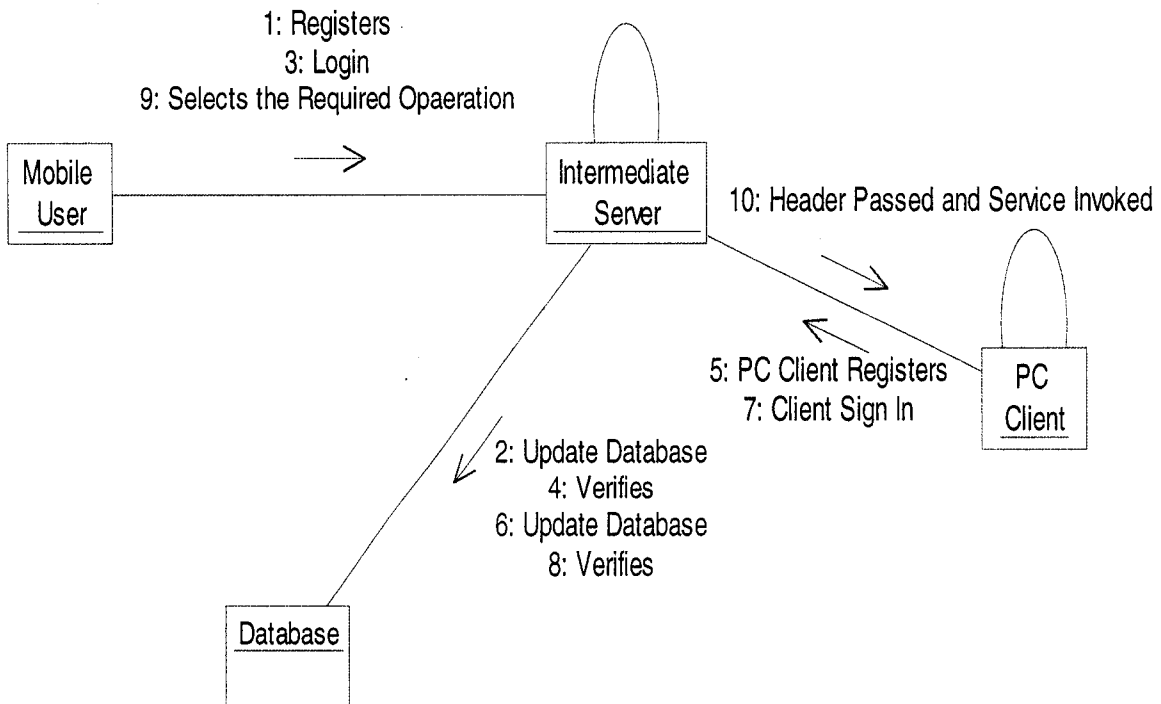


Figure 3.4.4 Collaboration Diagram

4. SOFTWARE IMPLEMENTATION MODEL

4.1 Module Description:

There are 5 modules in this system. They are:

- ◇ Mobile - Server Connection (J2ME)
- ◇ Client User Interface Design (VB)
- ◇ Intermediate Server Connection (Java)
- ◇ Invoking Admin Services (J2ME-Java-VB)

4.1.1 Mobile Server Connection:

The interface is established between the mobile and the intermediate server. Initially the login screen is displayed where the user has to enter the username and password .Both are then passed on to the server and verified with the information in the database.

After the username and password is authenticated, the screen containing the list of services that can be invoked is displayed.

4.1.2 Client User Interface Design:

The client initially gets connected to the intermediate server by registering to it. In case of an existing user, connection to the intermediate server is established when the user logs in.

The connection to the intermediate server is acquired with the help of Winsock control. The necessary data is passed with the help of socket programming to the server

4.1.3 Intermediate Server Connection:

The intermediate server is connected both to the client side as well as the mobile user. The connection to the server from the mobile user is established using socket programming in java and the connection from the client side to the server is done with the help of Winsock control

Role of the intermediate server:

- Verification of the username and password of the mobile user with the database.
- Passing the header associated with the corresponding commands from the mobile user to the client side.
- To pass the information about the user currently accessing the PC Client back to the mobile user.

4.1.4 Invoking Admin Services:

The following Admin services are invoked in the project undertaken:

- File Open
- Log Off
- Shut Down
- User ID
- Services
 - Apache Tomcat
 - Telnet
 - Themes
 - Telephony

FILE OPEN:

A header named “OPN” will be associated with the File Open command. As soon as the command is selected from the mobile user the header is passed from the mobile to the intermediate server and then passed on to the client. The program running at the client side recognizes the header and invokes the corresponding service. In this case the **shell** command is invoked that would open the file from the location and display it to the user at the client side

LOG OFF:

Here the header called “LOG” is passed on from the mobile user to the intermediate server and finally to the client side where the program recognizes the header and invokes the log off function. All the currently running programs will be terminated and the system logs off the current user automatically.

SHUT DOWN:

The header associated with this command at the mobile user side is “SHT”. As soon as the command is invoked from the header is passed to the intermediate server and then finally passed to the VB client side. The header is recognized at the client side and the corresponding function is invoked. All the programs thus terminate and the system shuts down automatically.

USER ID:

This service is used to get the information about the user who is currently using the system. When the service is invoked at the mobile user side the header that is associated with the service is (UID) is passed to the

intermediate server and passed to the client side using a socket in java. As the header is recognized at the client side, a function is called that retrieves the user information and passed back to the server. The user information is passed back to the intermediate server using the Winsock control .The information required is passed from the server back to the J2ME interface using sockets. The necessary information is then displayed using the textbox.

SERVICES:

The services that invoked in this project are:

- Apache Tomcat
- Telnet
- Themes
- Telephony

Apache Tomcat:

This administrator service is started and then stopped. The header associated with the service reaches the client side and then the corresponding service is invoked. The service can be started as well as stopped .Two different header's are associated with the two operations .When the operation is selected the header associated with it is passed and the service is invoked at the client side. When the particular service is started, operations in tomcat can be performed and it can be stopped when the service is not required.

Telnet:

Users can work and perform operations in telnet once it is started and can be stopped correspondingly when it is selected. The corresponding headers associated with them are passed to the client side through the intermediate server and then recognized at the VB Interface. When the header is identified the service is either started or stopped. Operations in Telnet can be invoked as the service is started.

Themes:

The themes can be invoked or deactivated according to the wish of the user using the PC currently. The header associated with the “Themes”: service is passed on to the client side through the intermediate server and it will be invoked if started or the default windows theme will be displayed when the service is stopped .The changes can be clearly noted when it is started or stopped.

Telephony:

This service provides Telephony API (TAPI) support for programs that control Telephony devices and IP based voice connections on the local computer and, through the LAN, on servers that are running on the service. This service can be invoked like all the above services when the corresponding header is passed to the client side from the mobile user through the intermediate server.

4.2 INPUT/OUTPUT DESIGN FORMATS:

4.2.1 Client side logging in

Input: username, password

Output: User is logged into the server.

4.2.2 Mobile User End

Input: username and password of the user

Output: Displays the list of services that can be controlled

5. PRODUCT TESTING

The system testing deals with the process of testing the system as a whole. This is done after the integration process. The entire system is tested by moving each module from top to bottom. The verification and validation process are being carried out. The errors that occur at the testing phase are eliminated and a well functioning system is developed.

5.1 TESTING METHODS

5.1.1 Unit Testing:

It focuses verification effort on the smallest unit of software design, the module. It is also known as module testing. The modules are tested separately. The testing is carried out during programming stage itself.

Each and every module is tested separately to check if its intended functionality is met. Some unit testing performed are,

- Checking the proper working of the system information and process list retrieval modules in the client application.
- Checking for proper connectivity between server and client and server and mobile.
- Ensuring that the messaging module works and does not interfere with other functions.

Verifying the integrity of the mobile's application interface and ensuring correctness of command transmission

5.1.2 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 reasonably expected by the users.

After a validation test has been conducted one of the two possible conditions exists:

- The function or the performance characteristics confirm to the specification and are accepted
- Deviation from the specification is uncovered and the deficiency list is created

5.1.3 Output Testing

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

5.1.4 Integration Testing

It is the testing performed to detect errors on interconnection between modules. Here, all the modules pertaining to the client system and server system are combined to form the client and server applications respectively and tested to ensure that they work in synchronization and without interference from each other.

6. FUTURE ENHANCEMENTS

The project Remote Admin Service Control is designed in such a way that future enhancements can be made in an easy manner. The project is quite flexible and can be easily customized according to the user's requirements.

One possible enhancement to the project is the control of additional services that are configured in the PC being worked in. This module would enable the mobile user to control other PC services using his mobile. The outputs of the commands executed would then be displayed on the mobile's screen.

Another useful enhancement is the addition of a screen capture module which would transmit a screenshot of the client's desktop to the administrator's mobile. This would better enable the administrator to monitor the clients in the network.

One another enhancement would be the transfer of the file contents that has been opened at the client side desktop, back to the mobile user end.

The mobile user could also be given the power to block or deny access to any user whom he feels is unauthorized to access his PC.

7. CONCLUSION

The project Remote Admin Service Control offers the following advantages:

- Allows the administrator to remain mobile while having total control of the administrator services of his PC.
- Quite flexible in nature allowing addition of newer service control more easily.
- Processing is done in the client side thus reducing the load on the server and reducing time taken.
- High security is present thus preventing misuse of the system.
- User friendly interface which requires no training.

8. APPENDIX

APPENDIX A - SAMPLE CODE

VB CLIENT SIDE CODE

Registration

```
Private Sub Form_Load()  
tcpClient.RemoteHost = "90.0.1.30"  
tcpClient.RemotePort = 1011  
tcpClient.Protocol = sckUDPProtocol  
tcpClient.Connect  
End Sub  
Private Sub Command1_Click()  
validate  
tcpClient.SendData (txtuname.Text + "*" + txtpswd.Text + "*" + a)  
Form1.Hide  
End Sub  
Public Function validate()  
With Me  
If (.txtuname.Text = "" Or .txtpswd.Text = "" Or .txtcpswd.Text =  
"" Or .txtemail.Text = "" Or .txtname.Text = "" Or .cboocc.Text =  
"") Then  
    MsgBox "Please enter all the required fields"  
ElseIf ((.txtpswd.Text) <> (.txtcpswd.Text)) Then  
    MsgBox " Please Re-enter the Password "  
    .txtpswd.Text = ""  
    .txtcpswd.Text = ""  
End With  
End Function
```

Else

 MsgBox " The user has been Successfully Registered!!!"

End If

End With

End Function

Services

Dim str As String

Private Sub Command1_Click()

 Winsock1.SendData (Text1.Text)

End Sub

Private Sub Form_Load()

 Form4.Width = 7000

 Form4.Height = 5000

 init

 Winsock1.RemoteHost = "90.0.1.30"

 Winsock1.RemotePort = 1022

 Winsock1.Protocol = skcUDPProtocol

 Winsock1.Connect

End Sub

Public Sub init()

 reqhand.Bind

End Sub

Private Sub reqhand_DataArrival(ByVal bytesTotal As Long)

 Dim head As String

 Dim str As String

```

Dim str1 As String
reqhand.GetData head
str = head
head = Left(head, 3)
If (head = "LOG") Then
    Shell ("LOGOFF.EXE")
End If
If (head = "SHT") Then
    Shell ("shutdown -s -t 00")
End If
If (head = "OPN") Then
    str = Right(str, (Len(str) - 4))
If Dir$(str) = "" Then
    MsgBox "Not file"
Else
    str1 = "c:\WINDOWS\notepad.exe " & str
    Shell (str1)
End If
End If
If (head = "TEL") Then
    Shell ("c:\b.bat")
End If

If (head = "TES") Then
    Shell ("c:\c.bat")
End If

```

```

If (head = "APA") Then
    Shell ("c:\d.bat")
End If
If (head = "APS") Then
    Shell ("c:\e.bat")
End If
If (head = "THE") Then
    Shell ("c:\f.bat")
End If
If (head = "THS") Then
    Shell ("c:\g.bat")
End If
If (head = "TPH") Then
    Shell ("c:\h.bat")
End If
If (head = "TPS") Then
    Shell ("c:\i.bat")
End If
If (head = "UID") Then
    str = getnames(True)
    Text1.Text = str
End If
End Sub
Function getnames(ByVal user_req As Boolean) As String "returns
    the computer name and user name
Dim m_name As String ' machine name

```

```

Dim u_name As String 'user name
Dim buf As String * 25 'user info buffer
Dim siz As Long
Dim x, y As Long
m_name = Space(16)
siz = Len(m_name)
x = GetComputerName(m_name, siz)
y = GetUserName(buf, 255)
u_name = Left(buf, InStr(buf, Chr(0)) - 1)
m_name = Trim(m_name)
m_name = Left(m_name, Len(m_name) - 1)
If user_req = True Then
    getnames = m_name & "*" & u_name & "*"
Else
If user_req = False Then
    getnames = m_name
End If
End If
End Function

```

MOBILE USER CODE

Code that passes password for verification

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.*;
import javax.microedition.io.*;
import java.io.*;

public class UDPInterface extends Thread{
    private String req;
    private String reply;
    private RemoteSerMIDLet parent;
    private Display display;
    private Form frm;
    private Alert a;
    Datagram dg;
    boolean b;
    byte[] bytes;

    public UDPInterface(RemoteSerMIDLet p,StringBuffer request) {
        req=request.toString();
        parent=p;
        frm=new Form("");
        display=Display.getDisplay(parent);
    }

    public void run(){
        b =sendRequest();
    }
}
```

```

if((b==true) && reply.equals("PWDOK"))
    {
        parent.authen_ok();
    }
else
    {
        a=new Alert("Authentication Failed");
        display.setCurrent(a);
    }
}

public boolean sendRequest(){
    try{
        DatagramConnection dc=(DatagramConnection)
        Connector.open("datagram://localhost:4444");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        dc.send(dg);
        dg=dc.newDatagram(1000);
        dc.receive(dg);
        dc.close();
        reply=new String(dg.getData(),0,dg.getLength());
        return true;
    }
}

```

```
        catch(Exception e){
            return false;
        }
    }
}
```

Code that passes the request to the server

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.*;
import javax.microedition.io.*;
import java.io.*;

public class reqhandler extends Thread
{
    String req;
    String reply;
    boolean b;
    byte[] bytes;
    Datagram dg;
    DatagramConnection dc;
    String filename;
    public reqhandler() {
        try{
            dc=(DatagramConnection)
            Connector.open("datagram://localhost:5555");
```



```

    }
    catch(Exception e)
    {
    }
}

public void run()
{
}

public void logoff(){
    try{
        req= new String("LOG");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        dc.send(dg);
        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}

public void shutdown(){
    try{
        req= new String("SHT");
        bytes=new byte[1000];

```

```

        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}

public void fileopen(String file){
    try{
        req= new String("OPN");
        filename=file;
        req=req.concat("*");
        req=req.concat(filename);
        System.out.println(req);
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("file sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
}

```

```

        catch(Exception e){
            //return false;
        }
    }

```

```

public void telnet(){
    try{
        req= new String("TEL");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
    catch(Exception e)
    {
    }
}

```

```

public void telnets(){
    try{
        req= new String("TES");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
    }
}

```

```

        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }

catch(Exception e){
    }
}

public void apache(){
    try{
        req= new String("APA");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}
}

```

```
public void apaches(){
    try{
        req= new String("APS");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}
```

```
public void themes(){
    try{
        req= new String("THE");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
    }
}
```

```

        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}

public void themess(){
    try{
        req= new String("THS");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}

public void telephony(){
    try{
        req= new String("TPH");
        bytes=new byte[1000];

```

```

        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}

public void telephonys(){
    try{
        req= new String("TPS");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        System.out.println("rec");
        dg=dc.newDatagram(1000);
    }
    catch(Exception e){
        //return false;
    }
}

```

```
public String user(){
    try{
        req= new String("UID");
        bytes=new byte[1000];
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        System.out.println("sent");
        dc.send(dg);
        dg=dc.newDatagram(1000);
        dc.receive(dg);
        dc.close();
        reply=new String(dg.getData(),0,dg.getLength());
        System.out.println("rec");
        System.out.println(reply);
        return reply;
    }
    catch(Exception e){
        reply="false";
        return reply;
    }
}
```


SERVER SIDE CODE

Code that handles the password

```
import java.io.*;
import java.net.*;
import java.util.StringTokenizer;
public class Worker{
static DatagramSocket sock;
static DatagramPacket p;
static String str,head,req,uname,pwd,addr;
static boolean b,reply;
public static void main(String a[]) throws Exception
{
DataInputStream in= new DataInputStream(System.in);
String ss;
sock=new DatagramSocket(4444);
p=new DatagramPacket(new byte[1000],1000);
sock.receive(p);
addr=p.getAddress().toString();
System.out.println("packet received From " + addr);
str=new String(p.getData(),0,p.getLength());
parseRequest();
b=checkRequest();
if(b==true){
String reply="PWDOK";
byte[] bytes=reply.getBytes();
```

```

        p=new
DatagramPacket(bytes,bytes.length,p.getAddress(),p.getPort());
        System.out.println("Press return to accept");
        ss=in.readLine();
        sock.send(p);
        System.out.println("Request from : " + addr
+"\\nRequest Accepted.. Connected");
    }
else
    {
        String reply="PWDFAILED";
        byte[] bytes=reply.getBytes();
        p=new
DatagramPacket(bytes,bytes.length,p.getAddress(),p.getPort());
        System.out.println("Press return to accept");
        ss=in.readLine();
        sock.send(p);
        System.out.println("Request from : " + addr +"\\nBad
authentication..Request rejected");
    }
}

public static void parseRequest() throws Exception
{
    head=str.substring(0,3);
    req=str.substring(3,str.length());
    StringTokenizer s=new StringTokenizer(req,"*");

```



```
uname=s.nextToken();
pwd=s.nextToken();
}
public static boolean checkRequest() throws Exception{
    if(head.equals("PWD"))
        {
            Database d=new Database(uname,pwd);
            reply=d.check();
            if(reply){
                return true;
            }
            else{
                return false;
            }
        }
    else{
        return false;
    }
}
}
```

Code that handles the request

```
import java.io.*;
import java.net.*;
import java.util.StringTokenizer;
public class mobhandle{
    static DatagramSocket sock;
    static DatagramPacket p;
    static String str,addr,msg,fname,req,head;
    public static void main(String a[]) throws Exception
    {
        DataInputStream in= new DataInputStream(System.in);
        String ss;
        sock=new DatagramSocket(5555);
        p=new DatagramPacket(new byte[1000],1000);
        sock.receive(p);
        System.out.println("packet received");
        str=new String(p.getData(),0,p.getLength());
        System.out.println(str);
        head=str.substring(0,3);

        if(head.equals("LOG")==true)
        {
            DatagramSocket sock=new DatagramSocket();
            byte buf[]=new byte[3000];
            InetAddress addr=InetAddress.getByName(null);
            buf=str.getBytes();
```

```

        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("logoff-data send");
    }
if(head.equals("SHT")==true)
    {
        DatagramSocket sock=new DatagramSocket();
        byte buf[]=new byte[3000];
        InetAddress addr=InetAddress.getByName(null);
        buf=str.getBytes();
        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("shutdown-data send");
    }
if(head.equals("OPN")==true)
    {
        req=str.substring(3,str.length());
        StringTokenizer s=new StringTokenizer(req,"*");
        fname=s.nextToken();
        System.out.println(fname);
        DatagramSocket sock=new DatagramSocket();
        byte buf[]=new byte[3000];
        InetAddress addr=InetAddress.getByName(null);
        buf=str.getBytes();
        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("file sent");
    }

```

```

if(head.equals("TEL")==true)
{
    DatagramSocket sock=new DatagramSocket();
    byte buf[]=new byte[3000];
    InetAddress addr=InetAddress.getByName(null);
    buf=str.getBytes();
    sock.send(new DatagramPacket(buf,buf.length,addr,9898));
    System.out.println("Telnet Invoked");
}

if(head.equals("TES")==true)
{
    DatagramSocket sock=new DatagramSocket();
    byte buf[]=new byte[3000];
    InetAddress addr=InetAddress.getByName(null);
    buf=str.getBytes();
    sock.send(new DatagramPacket(buf,buf.length,addr,9898));
    System.out.println("Telnet Terminated");
}

if(head.equals("APA")==true)
{
    DatagramSocket sock=new DatagramSocket();
    byte buf[]=new byte[3000];
    InetAddress addr=InetAddress.getByName(null);

```

```

        buf=str.getBytes();
        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("Apache Started");
    }
if(head.equals("APS")==true)
    {
        DatagramSocket sock=new DatagramSocket();
        byte buf[]=new byte[3000];
        InetAddress addr=InetAddress.getByName(null);
        buf=str.getBytes();
        DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("Apache Stoped");
    }
if(head.equals("THE")==true)
    {
        DatagramSocket sock=new DatagramSocket();
        byte buf[]=new byte[3000];
        InetAddress addr=InetAddress.getByName(null);
        buf=str.getBytes();
        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("Themes Invoked");
    }
if(head.equals("THS")==true)
    {
        DatagramSocket sock=new DatagramSocket();
        byte buf[]=new byte[3000];

```

```

        InetAddress addr=InetAddress.getByName(null);
        buf=str.getBytes();
        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("Themes Stopped");
    }
    if(head.equals("TPH")==true)
    {
        DatagramSocket sock=new DatagramSocket();
        byte buf[]=new byte[3000];
        InetAddress addr=InetAddress.getByName(null);
        buf=str.getBytes();
        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("Telephony Started");
    }
    if(head.equals("TPS")==true)
    {
        DatagramSocket sock=new DatagramSocket();
        byte buf[]=new byte[3000];
        InetAddress addr=InetAddress.getByName(null);
        buf=str.getBytes();
        sock.send(new DatagramPacket(buf,buf.length,addr,9898));
        System.out.println("Telephony Stopped");
    }

```



```
if(head.equals("UID")==true)
{
    DatagramSocket sock=new DatagramSocket();
    byte buf[]=new byte[3000];
    String request;
    InetAddress addr=InetAddress.getByName(null);
    buf=str.getBytes();
    sock.send(new DatagramPacket(buf,buf.length,addr,9898));
    UserId u=new UserId();
    u.start();
    DatagramPacket pac=u.Reuserid();
    request= new String(pac.getData(),0,pac.getLength());
    buf=new byte[1000];
    buf=request.getBytes();
    sock.send(new
    DatagramPacket(buf,buf.length,p.getAddress(),p.getPort()));
    System.out.println(request);
    System.out.println("ID passed");
}
}
}
```

APPENDIX B - SAMPLE OUTPUT

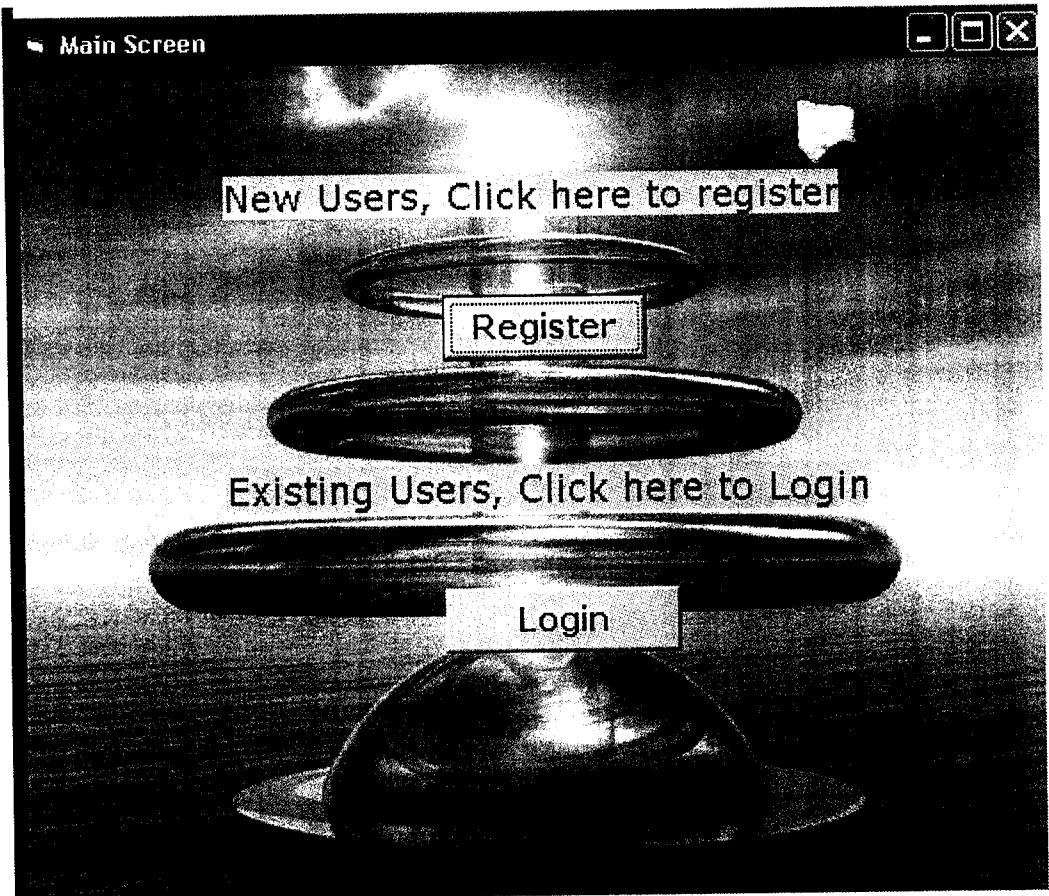


FIGURE 8.1 MAIN SCREEN

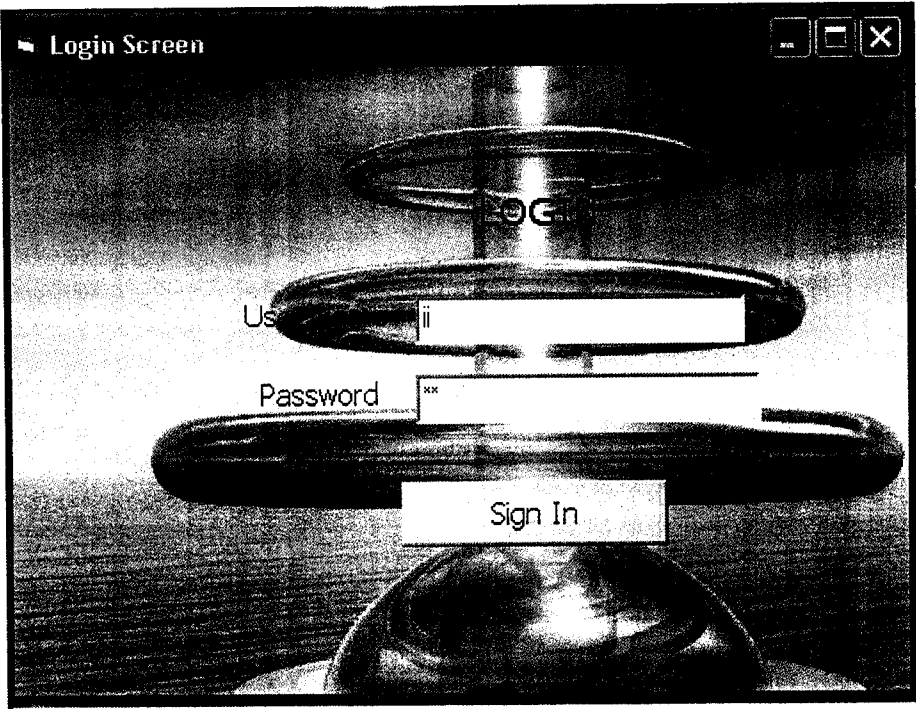


FIGURE 8.2 LOGIN SCREEN

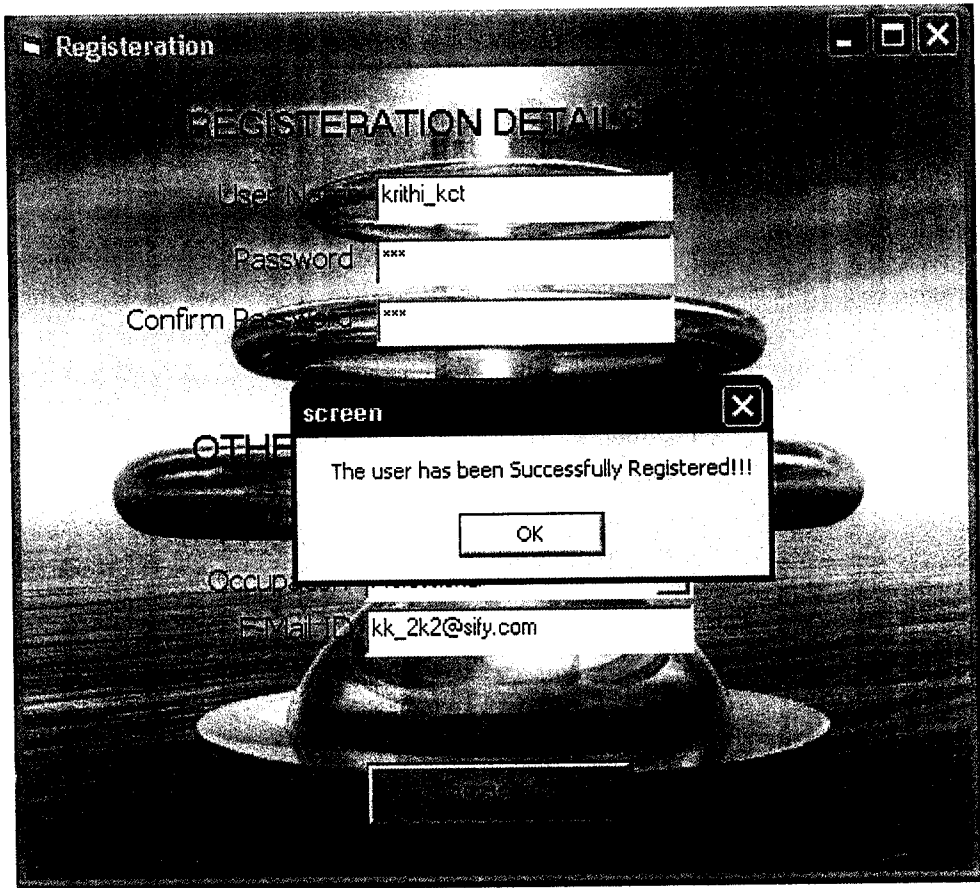


FIGURE 8.3 REGISTRATION FORM

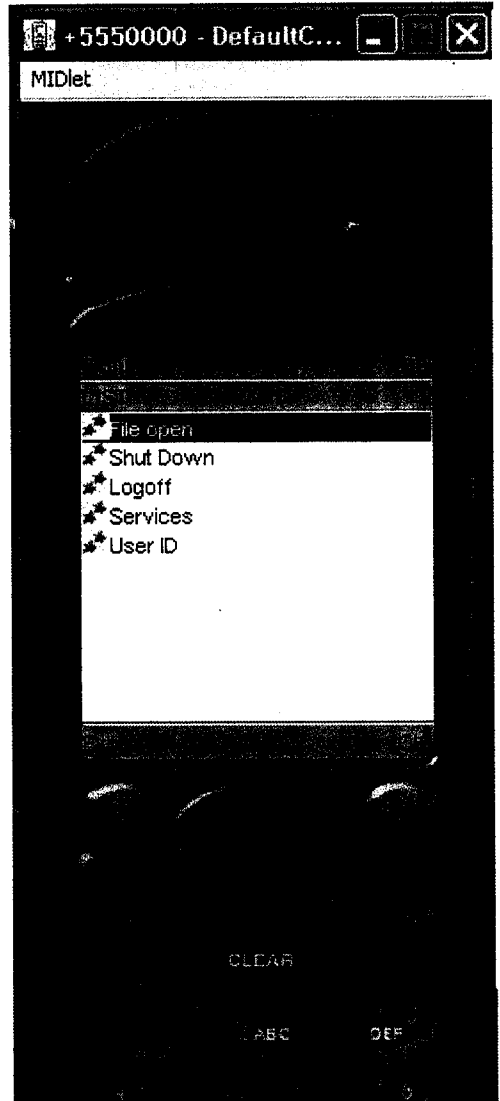
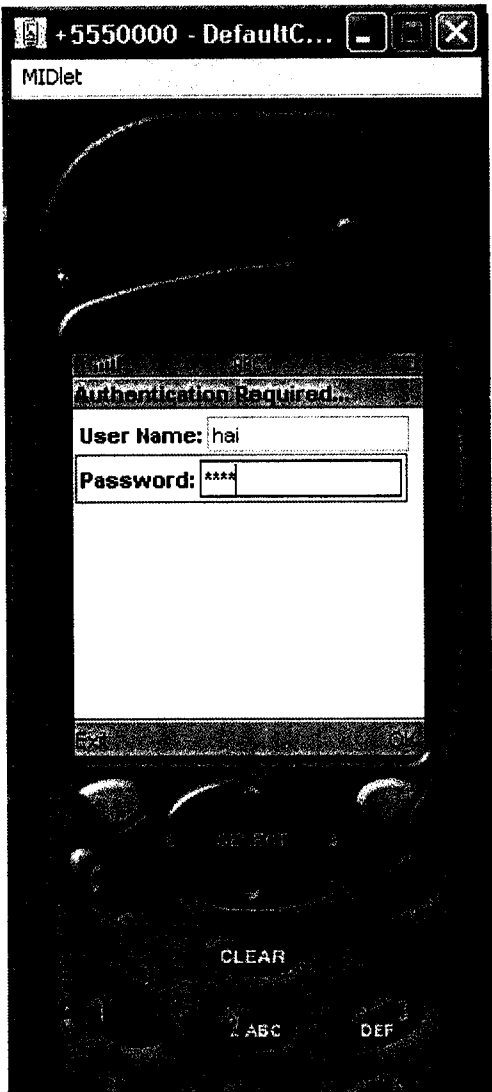


FIGURE 8.4

MOBILE USER SCREEN

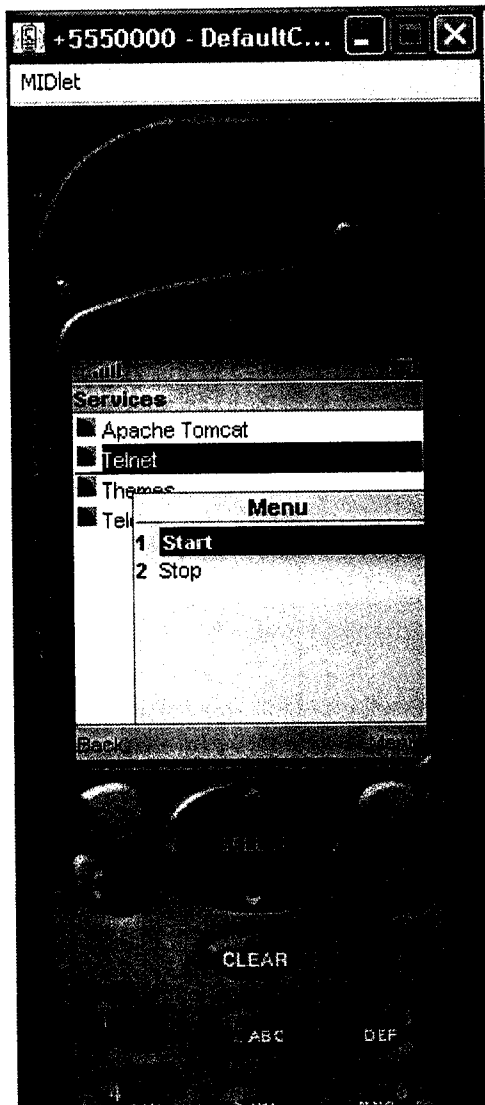


FIGURE 8.5 DISPLAY OF SERVICES

9. REFERENCES

Books:

1. Francesco Balena," Programming Microsoft Visual Basic 6.0",
Microsoft Press, 2001
2. Jonarthan Knudson," Wireless Java Programming with J2ME 2.0"

Website:

www.sun.java.com

www.java.sun.com/j2me/documentation.html

