P - 1643

NETWORK ADMINISTRATION THROUGH MOBILE DEVICES

A PROJECT REPORT

Submitted by

BALAJI.R	**71202205006**
DEEPAK.A	**71202205009**
HUSSAIN M PATEL	**71202205016**

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

MAY 2006

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"NETWORK ADMINISTRATION THROUGH MOBILE DEVICES"** is the bonafide work of "BALAJI.R, DEEPAK.A, HUSSAIN M PATEL " who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.G.GOPALSAMY	Mr.K.R.BASKARAN, M.S.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Information Technology	Asst. Professor
Kumaraguru College of Technology	Information Technology
Coimbatore-641006.	Kumaraguru College of Technology
	Coimbatore-641006.

The candidates with University Register Nos. **71202205006, 71202205009, 71202205016** were examined by us in the project viva-voce examination held on 26.04.06.

INTERNAL EXAMINER **EXTERNAL EXAMINER** 26/4/06

DECLARATION

We

R.BALAJI	**71202205006**
A.DEEPAK	**71202205009**
HUSSAIN M PATEL	**71202205016**

Declare that the project entitled **"NETWORK ADMINISTRATION THROUGH MOBILE DEVICES"**, 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 the guidance of **Mr.K.R.Baskaran, M.S., Asst.Prof.,** Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore
Date: 26/04/06.

[R.BALAJI]

[A.DEEPAK]

[HUSSAIN M PATEL]

Project Guided by

[Mr.K.R.Baskaran]

ACKNOWLEDGEMENT

We are extremely grateful to **Dr.K.K.Padmanabhan**, B.Sc. (Engg.)., M.Tech., Ph.D., Principal, Kumaraguru College Of Technology for having given us a golden opportunity to embark on this project.

We are deeply obliged to **Dr.G.Gopalsamy**, Head of the Department of Information Technology for his valuable guidance and useful suggestions during the course of this project.

We are indebted to our project guide **Mr.K.R.Baskaran**, M.S., Department of Information Technology for his helpful guidance and valuable support given to us throughout this project.

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

We also thank our friends and family who helped us to complete this project successfully.

ABSTRACT

This project is about mobilizing the network administration tasks, through mobile devices such as mobile phones, PDAs etc. The purpose of the project is to provide off-site network administration facilities through a simple mobile device. Remote administration is presently done using laptops and personal computers. Our project aims to mobilize the network administration.

The administrator accesses the server using the mobile device and carries out the administration tasks. The mobile application acts as an interface between the administrator and the network. The mobile application is hosted on the server side. The client machines are connected to the server in a network. The administrator carries out the tasks using the mobile device.

The project is mainly composed of three modules. They are mobile application, server side windows service and client side windows service. The various administration tasks that can be carried out are User Management, Group Management, Service Management, Event Log Viewing and Client Management.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CHTML	- Compact Hyper Text Markup language
HTML	- Hyper Text Markup Language
HTTP	- Hyper Text Transfer Protocol
IIS	- Internet Information Server
MMIT	- Microsoft Mobile Internet Toolkit
PDA	- Personal Desktop Assistant
WAP	- Wireless Application Protocol
WML	- Wireless Markup Language

1. INTRODUCTION

The main aim of this project is to create a system that helps the network administrator to control his network without being actually present physically at the network site. This administration process is designed to include system information gathering, process monitoring, session logging, and includes control actions and functions to give the administrator total control over all the clients in his network.

1.1 THE EXISTING SYSTEM AND ITS LIMITATIONS

As of now, no system exists which provides the functionality and mobility of the system designed. Conventional systems include functionalities such as network monitoring, controlling and client-server communication but fail to provide the administrator its flexibility to control his network even while remaining mobile. These systems require the administrator to remain physically at the server site.

LIMITATIONS

- Requires the administrator to be physically present at the network site.

1.2 THE 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 this system 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 network administrator 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. SYSTEM REQUIREMENTS ANALYSIS

The purpose of system requirements analysis is to obtain a detailed and thorough understanding of the business need. Then it is broken down into discrete requirements, which are then clearly defined, reviewed and agreed upon with the customers and decision makers. During system requirements analysis, the framework for the application is developed, providing the foundation for all future design and development efforts.

2.1 PRODUCT DEFINITION

Our intention is to develop a network administrative software which allows the administrator to control his network even while remaining mobile without the need to be physically present at the network site.

2.2 PROJECT PLAN

In the analysis phase, we learnt about various technologies that could be suitable for the implementation. From this study, the most suitable languages for implementing this project were found to be VB.NET and ASP.NET along with active directory support. Differentiation of modules, user interface design and other crucial aspects were identified and designed as necessary.

From the analysis phase, the design phase commences in which the various modules functionalities are identified. The complete system's flow of control and data are identified and depicted in the form of diagrams.

Next the implementation phase is taken care of in which the coding of the project is done. Each module is coded separately and finally integrated to form the entire application. Care is taken to make the code easily understandable by future users.

In the testing phase, each module is tested thoroughly and finally integrated modules are tested together to ensure the correct working of the entire application. Testing is also done to ensure this application satisfied the specified requirements and set criteria.

2.3 SOFTWARE REQUIREMENTS SPECIFICATION

2.3.1 PURPOSE

This Software Requirement Specification describes the function and performance requirements allotted to the "**Network Administration through Mobile Devices**". It is a mobile-based system designed to provide basic functions of network administration.

2.3.2 SCOPE

The major purpose is providing basic network administration function to a network administrator through user's mobile device. It has numerous benefits over the existing remote administration techniques

2.3.3 DEFINITIONS

Customer:
A person or organization, internal or external to the producing organization, who takes financial responsibility for the system. In a large system, this may not be the end user. The customer is the ultimate recipient of the developed product and its artifacts.

User:
A person who will use the system that is developed.

Analyst:
The analyst details the specification of the systems functionality by describing the requirements aspect under the supporting software requirements.

2.3.4 ABBREVIATIONS

VB.NET : Visual Basic .NET
SRS : Software Requirements Specification
MMIT : Microsoft Mobile Internet Toolkit
IIS : Internet Information Server

2.3.5 GENERAL DESCRIPTION

2.3.5.1 PRODUCT OVERVIEW

The purposed system will serve to replace the currently existing remote administration system that is widely used. Administrative functions will be listed by the mobile application. Based on the users choice appropriate message will be communicated to a windows service, which will perform the required administrative task. The external interface of this mobile system will be the mobile device's browser. This will work on any mobile devices regardless of its web access methods. This system will be able to render WML pages for WAP based devices, HTML; for PC browsers and cHTML for i-mode based mobile devices.

2.3.5.2 USER CHARACTERISTICS

- **NETWORK ADMINISTRATOR:** The user in the network who has full administrative rights and is responsible for the network administrative activities.
- **REMOTE ADMINISTRATOR:** A network administrator who accesses the mobile application to perform basic administration function.

2.3.5.3 GENERAL CONSTRAINTS

Since the mobile application development is based on .NET technologies, .NET framework should be preinstalled in the server as well as the clients, which are being controlled. MMIT should also be installed in the server as this is the package which provides the basic mobile controls, interface, etc.

2.3.6 SPECIFIC REQUIREMENTS

2.3.6.1 INPUTS AND OUTPUTS

The main input of the system is the administrator's login id and password which will be used to authenticate him. All functions are pre-defined and do not require any input from the administrator. The administrator just has to select from any of the pre-defined functions available to him, however, the administrator is required to input a message for the messaging function.

The expected output of the system is the total control of the network by the administrator and proper working of all the functions performed by him.

2.3.6.2 FUNCTIONAL REQUIREMENTS

1. The system should be able to authenticate the network administrator.
2. The system should allow the admin to perform the desired functions such as system information gathering, shutdown, logoff and process termination without any problem.

2.3.6.3 HARDWARE REQUIREMENTS

Processor	PENTIUM IV
RAM	128 MB
Hard Disk Capacity	20 GB

2.3.6.4 SOFTWARE REQUIREMENTS

Software	Net framework, IIS and MMTT
Operating System	Windows 2000 Server

2.3.6.5 PERFORMANCE CONSTRAINTS

The system runs smoothly as long as GPRS connection is available and returns results of any operation within a time limit of 10 seconds.

2.3.6.6 SOFTWARE CONSTRAINTS

The clients need .NET framework installed on any windows platform. The server requires .NET framework installed on Windows 2000 Server and running in the server system.

3. SYSTEM STUDY

3.1 INTRODUCTION TO .NET AND .NET FRAMEWORK

Microsoft® .NET is a set of Microsoft software technologies for connecting information, people, systems, and devices. It enables a high level of software integration through the use of Web services—small, discrete, building-block applications that connect to each other as well as to other, larger applications over the Internet.

NET is infused into the products that make up the Microsoft platform, providing the ability to quickly and reliably build, host, deploy, and utilize connected solutions using Web services, all with the protection of industry-standard security technologies.

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.

- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components:

1. **The common language runtime** and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code.

2. **The class library**, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

3.2 INTRODUCTION TO VB.NET

Visual Basic .NET (VB.NET) is an object-oriented computer language that can be viewed as an evolution of Microsoft's Visual Basic (VB) implemented on the Microsoft .NET framework. Its introduction has been controversial, as significant changes were made that broke backward compatibility with VB and caused a rift within the developer community that may or may not be resolved with the introduction of Visual Studio 2005.

The vast majority of VB.NET developers use Visual Studio .NET, although Sharp Develop provides an open-source alternative. The creation of open-source tools for VB.NET development have been slow compared to C#, although the Mono development platform provides an implementation of VB.NET-specific libraries and is working on a compiler, as well as the Windows Forms GUI library.

Similar to the virtual machine requirement of other programming languages, applications written in VB.NET require the .NET framework to execute.

Relation to Visual Basic

Whether Visual Basic .NET should be considered as just another version of Visual Basic or a completely different language is a topic of debate. This is not obvious, as once the methods that have been moved around and which can be automatically converted are accounted for, the basic syntax of the language has not seen many "breaking" changes, just additions to support new features like structured exception handling and short circuited expressions. One simple change that can be confusing to previous users is that of Integer and Long data types, which have each doubled in length; a 16-bit integer is known as a Short in VB.NET, while Integer and Long are 32 and 64 bits respectively. Similarly, the Windows Forms GUI editor is very similar in style and function to the Visual Basic form editor.

The things that have changed significantly are the semantics — from those of a procedural programming environment running on a deterministic, reference-counted engine based on COM to a fully object-oriented language backed by the .NET Framework, which consists of a combination of the Common Language Runtime (a virtual machine using generational garbage collection and a just-in-time compilation engine) and a far larger class library. The increased breadth of the latter is also a problem that VB developers have to deal with when coming to the language, although this is somewhat addressed by the My feature in Visual Studio 2005.

The changes have altered many underlying assumptions about the "right" thing to do with respect to performance and maintainability. Some functions

and libraries no longer exist; others are available, but not as efficient as the "native" .NET alternatives.

3.3 INTRODUCTION TO ASP.NET

ASP.NET is a set of web development technologies marketed by Microsoft. Programmers can use it to build dynamic web sites, web applications and XML web services. It is part of Microsoft's .NET platform and is the successor to Microsoft's Active Server Pages (ASP) technology.

Even though ASP.NET takes its name from Microsoft's old web development technology, ASP, the two differ significantly. Microsoft has completely rebuilt ASP.NET, based on the Common Language Runtime (CLR) shared by all Microsoft .NET applications. Programmers can write ASP.NET code using any of the different programming languages supported by the .NET Framework, usually (proprietary) Visual Basic.NET, JScript .NET, or (standardized) C#, but also including open-source languages such as Perl and Python. ASP.NET has performance benefits over other script-based technologies because the server-side code is compiled to one or a few DLL files on a web server.

ASP.NET attempts to simplify developers' transition from Windows application development to web development by offering the ability to build pages composed of controls similar to a Windows user interface. A web control, such as a button or label, functions in very much the same way as its Windows counterpart: code can assign its properties and respond to its events. Controls know how to render themselves: whereas Windows controls draw themselves to the screen, web controls produce segments of HTML and JavaScript which form part of the resulting page sent to the end-user's browser.

ASP.NET encourages the programmer to develop applications using an event-driven GUI paradigm, rather than in conventional web-scripting environments like ASP and PHP. The framework attempts to combine existing technologies such as JavaScript with internal components like "Viewstate" to bring persistent (inter-request) state to the inherently stateless web environment.

ASP.NET uses the .NET Framework as an infrastructure. The .NET Framework offers a managed runtime environment (like Java), providing a virtual machine with JIT and a class library.

The numerous .NET controls, classes and tools can cut down on development time by providing a rich set of features for common programming tasks. Data access provides one example, and comes tightly coupled with ASP.NET. A developer can make a page to display a list of records in a database, for example, significantly more readily using ASP.NET than with ASP.

Advantages of ASP.NET over ASP

- Compiled code means applications run faster with more design-time errors trapped at the development stage.
- Significantly improved run-time error handling, making use of exceptions and try-catch blocks.
- User-defined controls allow commonly used templates, such as menus.
- Similar metaphors to Windows applications such as controls and events, which make development of rich user interfaces, previously only found on the desktop, possible.
- An extensive set of controls and class libraries allows the rapid building of applications.

- ASP.NET leverages the multi-language capabilities of the .NET CLR, allowing web pages to be coded in VB.NET, C#, J#, etc.
- Ability to cache the whole page or just parts of it to improve performance.
- Ability to use the "code-behind" development model to separate business logic from presentation.
- If an ASP.NET application leaks memory, the ASP.NET runtime unloads the AppDomain hosting the erring application and reloads the application in a new AppDomain.
- Session state in ASP.NET can be saved in a SQL Server database or in a separate process running on the same machine as the web server or on a different machine. That way session values are not lost when the web server is reset or the ASP.NET worker process is recycled.

3.4 INTRODUCTION TO ACTIVE DIRECTORY

Active Directory (codename Cascade) is an implementation of LDAP directory services by Microsoft for use in Windows environments. Active Directory allows administrators to assign enterprise-wide policies, deploy programs to many computers, and apply critical updates to an entire organization. An Active Directory stores information and settings relating to an organization in a central, organized, accessible database. Active Directory networks can vary from a small installation with a few hundred objects, to a large installation with millions of objects.

Active Directory was previewed in 1996, released first with Windows 2000, and saw some revision to extend functionality and improve administration in Windows Server 2003.

Objects

An Active Directory (AD) structure is a hierarchical framework of objects. The objects fall into three broad categories — resources (e.g. printers), services (e.g. e-mail), and people (accounts, or users and groups). The AD provides information on the objects, organizes the objects, controls access, and sets security.

Each object represents a single entity — whether a user, a computer, a printer, an application, or a shared data source — and its attributes. Objects can also be containers of other objects. An object is uniquely identified by its name and has a set of attributes — the characteristics and information that the object can contain — defined by and depending on its type. The attributes, the basic structure of the object itself, are defined by a schema, which also determines the kind of objects that can be stored in the AD.

The schema itself is made up of two types of objects: schema class objects and schema attribute objects. A single schema class object defines one type of object that can be created by AD — for instance, it allows a User object to be created — and a schema attribute object defines an attribute that objects can have.

Each attribute object can be used in several different schema class objects. Those objects are known as schema objects, or metadata, and exist to allow the schema to be extended or modified when necessary. However, because each schema object is integral to the definition of AD objects, deactivating or changing these objects can have serious consequences because it will fundamentally change the structure of AD itself. A schema object, when altered, will automatically propagate through Active Directory and once it is

created it can only be deactivated — not deleted. Changing the schema is not something that is usually done without some planning

Forests, Trees and Domains

The framework that holds the objects is viewed at a number of levels. At the top of the structure is the Forest - the collection of every object, its attributes and rules (attribute syntax) in the AD. The forest holds one or more transitive trust linked Trees. A tree holds one or more Domains and domain trees, again, linked in a transitive trust hierarchy. Domains are identified by their DNS name structure, the namespace. A domain has a single DNS name.

The objects held within a domain can be grouped into containers called Organizational Units (OUs). OUs give a domain a hierarchy, ease its administration, and can give a semblance of the structure of the AD's company in organizational or geographical terms. OUs can contain OUs - indeed, domains are containers in this sense - and can hold multiple nested OUs. Microsoft recommends as few domains as possible in AD and a reliance on OUs to produce structure and improve the implementation of policies and administration. The OU is the common level at which to apply group policies, which are AD objects themselves called Group Policy Objects (GPOs), although policies can also be applied to domains or sites (see below). The OU is the lowest level at which administrative powers can be delegated.

As a further subdivision AD supports the creation of Sites, which are physical, rather than logical, groupings defined by one or more IP subnets. Sites distinguish between locations connected by low-speed (e.g. WAN, VPN) and high-speed (e.g. LAN) connections. Sites can contain one or more domains and domains can contain one or more sites. This is important to control network traffic generated by replication.

The actual division of the company's information infrastructure into a hierarchy of one or more domains and top-level OUs is a key decision. Common models are by business, by geographical location, or by IT roles. These models are also often used in combination.

4. DESIGN DOCUMENTS

4.1 INTRODUCTION

A software design is representation of the system of the real world in a format that can be easily understood by both the developers and the users. The diagrams drawn in software design help easy communication between the developers of the various modules of the system and with the users of the system.

In the object oriented paradigm which we have adopted to develop our system, it is easy to make such communication using a standard notation known as UML (Unified Modeling Language). It is an industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems standardized by the Object Management Group. UML simplifies the complex process of software design by using "blueprints" for software construction. Widespread adoption of UML is one of the forces contributing to developer demand for tools that can represent more intentional problem-related information.

4.2 PROCESS DESIGN

The design of a process in the object oriented paradigm can be represented using various UML diagrams like class diagram, sequence diagram, activity diagram, collaboration diagram, etc…

Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system. Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes. The various kinds of associations

that can exist between classes are generalization, specialization, composition, aggregation, etc.

A class diagram is a pictorial representation of the detailed system design. Design experts who understand the rules of modeling and designing systems design the system's class diagrams. The structure of a system is represented using class diagrams. Class diagrams are referenced time and again by the developers while implementing the system.

4.2.1 System Class Diagram

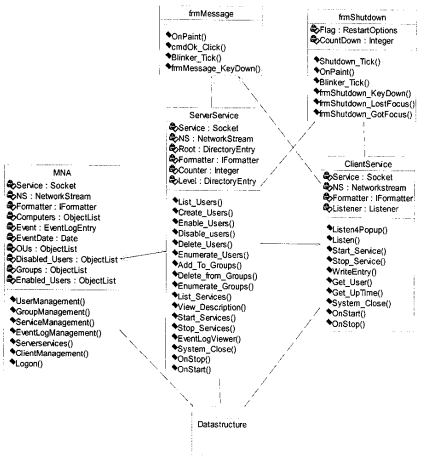


Fig .4.1 System Class Diagram

4.2.2 Data Structure Package

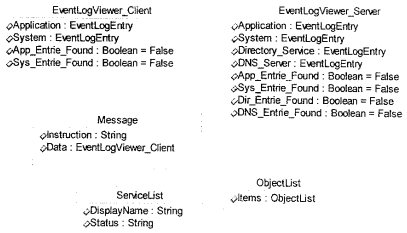


Fig.4.2 Data Structure Package

4.2.3. UseCase Diagram for Overall System

This UseCase diagram is used to depict the interactions in Overall System.

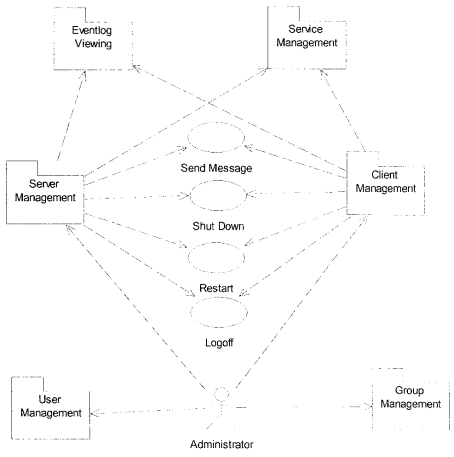


Fig. 4.3. UseCase Diagram for Overall System

4.2.4. UseCase Diagram for User Management

This UseCase diagram us used to depict the interactions in User Management.

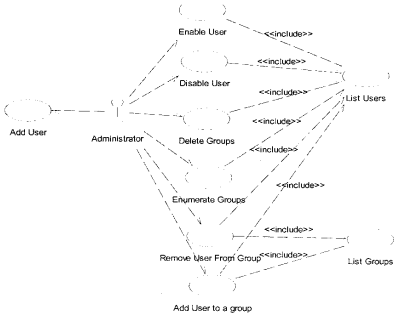


Fig.4.4. UseCase Diagram for User Management

4.2.5. UseCase Diagram for Group Management

This Usecase diagram is used to depict the interactions in Group Management.

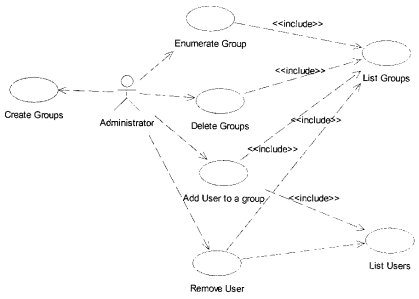


Fig.4.5. UseCase Diagram for Group Management

4.2.6. UseCase Diagram for Service Management

This Usecase diagram is used to depict the interactions in Service Management.

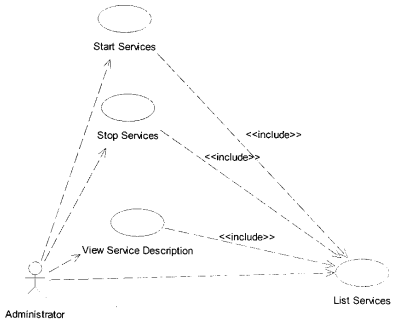


Fig.4.6. UseCase Diagram for Service Management

4.2.7. UseCase Diagram for Event Log Viewer

This Usecase diagram is used to depict the interactions in Event log Viewing.

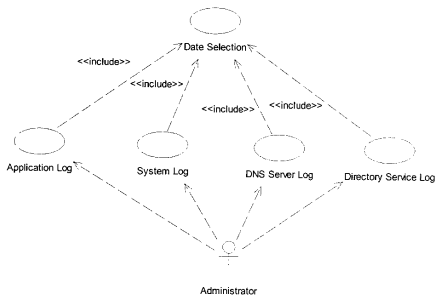


Fig.4.7. UseCase Diagram for Event Log Viewer

4.2.8. UseCase Diagram for Client Management

This Usecase diagram is used to depict the interactions in Client Management.

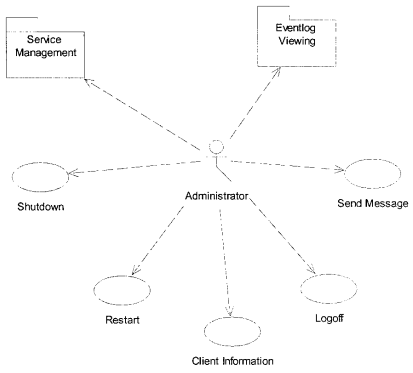


Fig. **4.8. UseCase Diagram for Client Management**

4.2.9. UseCase Diagram for Server Management

This Usecase diagram is used to depict the interactions in Service Management.

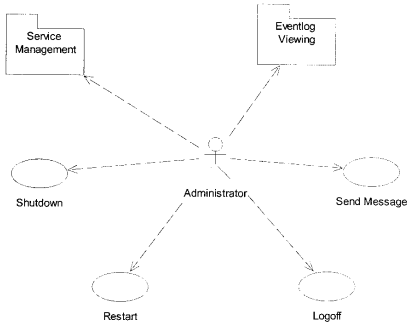


Fig.4.9. UseCase Diagram for Server Management

4.2.10. Sequence Diagram to Add User

This Sequence Diagram is used to depict the sequence of operations for adding Users.

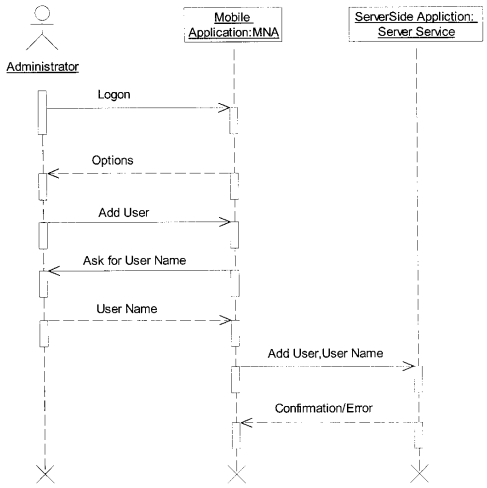


Fig. **4.10. Sequence Diagram to Add User**

4.2.11. Sequence Diagram to Add User to a Group

This Sequence Diagram is used to depict the sequence of operations for adding a user to groups.

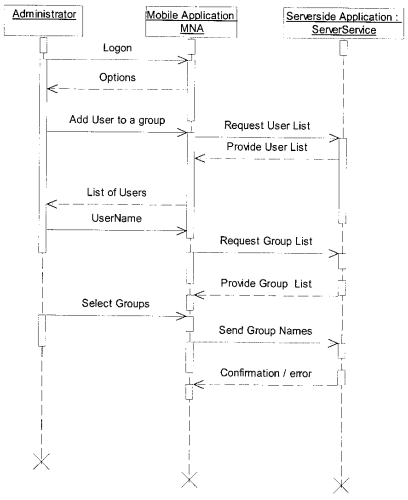


Fig.4.11. Sequence Diagram to Add User to a Group

4.2.12. Sequence Diagram to Disable User

This Sequence Diagram is used to depict the sequence of operations for disabling a user.

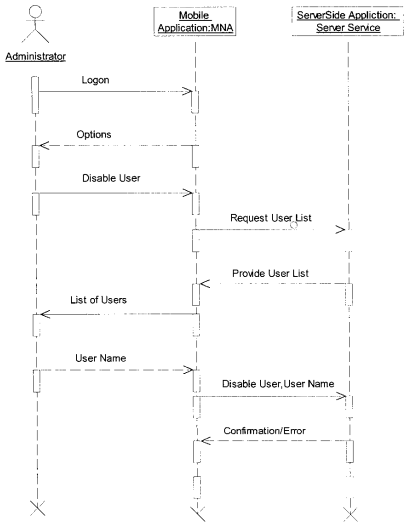


Fig.4.12. Sequence Diagram to Disable User

4.2.13. Sequence Diagram to Enable User

This Sequence Diagram is used to depict the sequence of operations for enabling a user.

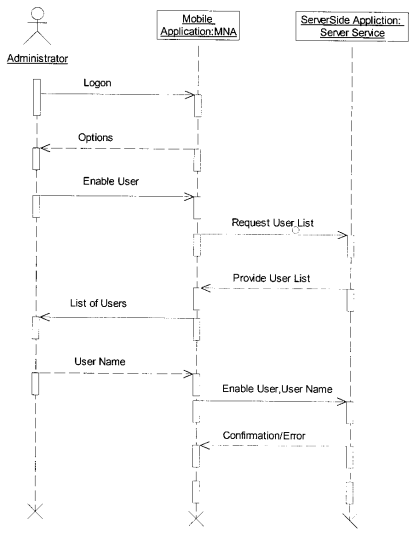


Fig.4.13. Sequence Diagram to Enable User

4.2.14. Sequence Diagram to Remove User from Groups

This Sequence Diagram is used to depict the sequence of operations for removing users from groups.

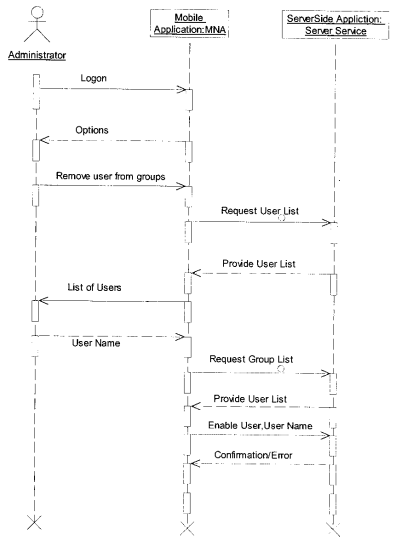


Fig.4.14. Sequence Diagram to Remove User from Groups

5. SOFTWARE IMPLEMENTATION MODEL

5.1 MODULE DESCRIPTION

The proposed system consists of the following modules
MODULE 1 :MOBILE APPLICATION
MODULE 2 :SERVER SIDE WINDOWS SERVICE
MODULE 3:CLIENT SIDE WINDOWS SERVICE

5.2 MOBILE APPLICATION

This module involves the implementation of the mobile application. With the use of this mobile application we can interact with the server. The mobile application can be accessed through any mobile phone, PDA etc. The mobile application is hosted on the server side. The network administrator can thus access the mobile application through the widely used HTTP protocol via any mobile web browser.

The user is presented with a login interface for authentication. The mobile application then communicates with the server side windows service and authenticates the user against the 'Active Directory'. On being successfully authenticated, the user can then manage the network through the application. This application communicates the requests of the user to the server side windows service which performs the necessary tasks for the user.

The module is responsible for mobile independence of the system. It is capable of rendering HTML, WML and cHTML pages. So this mobile application can be used by any WAP based mobile devices, PDAs and i-

mode based devices. The rendering of web pages is carried out according to the header information present in each request. This header information is used to identify the device which is making the request.

5.3 SERVER SIDE WINDOWS SERVICE

The server side application includes usage of the Active directory service. Active Directory is the directory service for Windows 2000 Server. It stores information about objects on the network and makes this information easy for administrators and users to find and use. Active Directory service uses a structured data store as the basis for a logical, hierarchial organization of directory information. Security is integrated with Active Directory through logon authentication and access control to objects in the directory.

With a single network logon, administrators can manage directory data and organization throughout their network, and authorized network users can access resources anywhere on the network. Policy-based administration eases the management of even the most complex network. A domain defines a security boundary. The directory includes one or more domains, each having its own security policies and trust relationships with other domains. Domains provide several benefits.

Domains help structure your network to better reflect your organization. Each domain stores only the information about the objects located in that domain. By partitioning the directory this way, 'Active Directory' can scale to very large numbers of objects.

This is the backbone of the system. It is the one that is responsible for performing the tasks requested by the user. This is achieved by accessing the 'Active Directory' through the 'DirectoryEntry' class present in the system.DirectoryServices namespace.

5.4 CLIENT SIDE WINDOWS SERVICE

This is installed on every client present in the domain. This service is responsible for performing the client side activities like System logoff, restart, shutdown etc... and also activities like retrieving system information, service management, etc...

6. 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 traversing 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.

6.1 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 seperately. The testing is carried out usually during programming stage itself.

Each and every module is tested seperately to check if its intended functionality is met. Some unit testing tasks performed are listed below.

- Checking the proper working of the system information and process list retrieval modules in the client application.
- Checking for proper connectivity between server – client and server-mobile.
- Verifying the integrity of the mobile's application interface and ensuring correctness of command transmission.
- Ensuring that the messaging module works and does not interfere with other functions.

6.2 VALIDATION TESTING

Validation is a process of finding out if the product being built is right, i.e. whatever the software product is being developed, and it should do what the user expects it to do. The software product should functionally do what it is supposed to, it should satisfy all the functional requirements set by the user. Validation is done during or at the end of the development process in order to determine whether the product satisfies specified requirements.

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

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

6.3 OUTPUT TESTING

After performing the validation testing, the next step is the 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 and displayed by the system under consideration are tested by the users about the formats required by them.

6.4 INTEGRATION TESTING

The testing of related modules, program, validates that multiple parts of the system Interact according to the plan. The three modules in our project are integrated, such that every functions exposed by the mobile application are

carried out exactly as per the requirements, by the server side windows service and the client side windows service.

6.5 SYSTEM TESTING

The system is tested against the software requirements specification to see if all the requirements are met and if the system performs as per the client's expectations. The system is tested as a whole to check for its functionality. Non functional requirements like performance considerations and platform support are checked as a whole.

7. FUTURE ENHANCEMENTS

The windows service modules can be extended to provide extended functionalities to enable better administration. Advanced network services can be implemented. The single user system can further be extended to feature multiple user access.

8. CONCLUSION

The network administrator plays an important role in the administration of any network. It is imperative that the administrator is present at the location of the network. The administrator is responsible for all the tasks like managing users, their privileges, monitoring the network etc.

Our project facilitates the administrator to remotely administer the network when he is not present at the location of the network. Our project provides features to mobilize the administration. The objective of this project is to put forward the problems faced in remote administration and to propose a viable solution. It proposes a solution to provide basic network administrative functions through any mobile device that a network administrative may have. We mainly concentrate only on the core or the essential administrative tasks like account management, group management, etc.

9. APPENDIX

9.1 SAMPLE SOURCE CODE

Mobile :

```
Imports System.Net
Imports System.IO
Imports System.ServiceProcess
Imports System.Net.Sockets
Imports System.Runtime.Serialization
Imports System.Runtime.Serialization.Formatters.Binary
Imports System.Diagnostics
Imports DataStructure
Public Class MNAClass
    Inherits System.Web.UI.MobileControls.MobilePage
    Shared Service As Socket = Global.Service
    Shared NS As NetworkStream = Global.NS
    Shared Formatter As IFormatter = Global.Formatter
    Shared Disabled_Users, Enabled_Users, Users, Group_Members,
Client_Info As ObjectList
    Shared Computers, Groups, OUs, GroupMembership As ObjectList
    Shared Server_Services, Client_Services As New ServiceList
    Shared EventLog_Server As EventLogViewer_Server
    Shared EventLog_Client As EventLogViewer_Client
    Shared LastDate_Server, LastDate_Client, LastSystem As String
    Shared Entries() As EventLogEntry
    Shared EventDate As Date
    Shared Position As Integer

    Protected WithEvents cmdSignIn As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdUser As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdClient As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdEventLog As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdService As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdListUser As
System.Web.UI.MobileControls.Command
```



```
    Protected WithEvents cmdDeleteUser As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdDisableUser As
System.Web.UI.MobileControls.Command
    Protected OUListView As System.Web.UI.MobileControls.SelectionList
    Protected UserListView As System.Web.UI.MobileControls.List
    Protected txtUser As System.Web.UI.MobileControls.TextBox
    Protected Label6 As System.Web.UI.MobileControls.Label
    Protected WithEvents cmdEnableUser As
System.Web.UI.MobileControls.Command
    Protected lblError As System.Web.UI.MobileControls.Label
    Protected WithEvents cmdSelect_User As
System.Web.UI.MobileControls.Command
    Protected UserSelectionList As
System.Web.UI.MobileControls.SelectionList
    Protected WithEvents Command1 As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdListGroups As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdCreateGroups As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdDeleteGroups As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdAddUsersToGroup As
System.Web.UI.MobileControls.Command
    Protected WithEvents cmdEnumerateUsers As
System.Web.UI.MobileControls.Command
    Protected GroupListView As System.Web.UI.MobileControls.List
    Protected GroupSelectionList As
System.Web.UI.MobileControls.SelectionList
    Protected WithEvents cmdSelect_Group As
System.Web.UI.MobileControls.Command
    Protected Label10 As System.Web.UI.MobileControls.Label
    Protected txtGroupName As System.Web.UI.MobileControls.TextBox
    Protected WithEvents cmdAddGroup As
System.Web.UI.MobileControls.Command
    Protected TypeSelection As
System.Web.UI.MobileControls.SelectionList
    Protected OptionSelection As
System.Web.UI.MobileControls.SelectionList
    Protected WithEvents cmdLogoff As
System.Web.UI.MobileControls.Command
    Protected frmError As System.Web.UI.MobileControls.Form
```

```
  Protected WithEvents cmdBack_frmError As
System.Web.UI.MobileControls.Command
  Protected WithEvents cmdRemoveUsersFromGroup As
System.Web.UI.MobileControls.Command
  Protected WithEvents cmdAddToGroups As
System.Web.UI.MobileControls.Command
  Protected WithEvents cmdRemoveFromGroups As
System.Web.UI.MobileControls.Command
  Protected WithEvents cmdEnumerateGroup As
System.Web.UI.MobileControls.Command

#Region " Web Form Designer Generated Code "

  'This call is required by the Web Form Designer.
  <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
    Me.KeepAlive = New System.Timers.Timer
    CType(Me.KeepAlive,
System.ComponentModel.ISupportInitialize).BeginInit()
    '
    'KeepAlive
    '
    Me.KeepAlive.Interval = 30000
    CType(Me.KeepAlive,
System.ComponentModel.ISupportInitialize).EndInit()

  End Sub

  Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
    'CODEGEN: This method call is required by the Web Form Designer
    'Do not modify it using the code editor.
    InitializeComponent()
  End Sub

#End Region

  Private Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    If Not IsPostBack Then
      LastDate_Client = ""
      LastDate_Server = ""
      LastSystem = ""
```



```
    End If
    If String.Equals(Session("Logged"), "True") Then
      KeepAlive.Enabled = True
    End If
  End Sub

  Private Sub cmdSignIn_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdSignIn.Click
    ActiveForm = frmLogin
  End Sub

  Function Wait4Data() As Boolean
    Dim Start As Date = Now
    While NS.DataAvailable = False
      If DateDiff(DateInterval.Minute, Start, Now) > 1 Then
        Return True
      End If
    End While
    Return False
  End Function

  Sub Request_TimedOut()
    Response.Write("<H3><CENTER><FONT color=red>Request Timed
Out!</FONT></CENTER></H3><br>")
    Response.Write("<B><Center>The server has not responded for a
long time! Try to sign-in again!</Center></B><br><br>")
    Response.Write("<b><center><font color=blue><a
href=http://Server/MNA/Default.aspx> <u>Login
Again</u></a></font></center></b>")
    KeepAlive.Enabled = False
    Try
      NS.Close()
      Service.Shutdown(SocketShutdown.Both)
      Service.Close()
    Catch ex As Exception
    End Try
    Session("Logged") = "False"
    Session.Clear()
    Session.Abandon()
    ActiveForm = frmLoggedOff
  End Sub
```



```
  Private Sub frmOUList_Activate(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles frmOUList.Activate
    If String.Equals(Session("OUs_Listed"), "False") Then
      Dim Command As New Message
      Command.Instruction = "List OUs"
      Formatter.Serialize(NS, Command)

      If Wait4Data() Then
        Request_TimedOut()
        Exit Sub
      End If

      OUs = DirectCast(Formatter.Deserialize(NS), ObjectList)
      Session("OUs_Listed") = "True"
    End If

    OUListView.Items.Clear()
    For Each Item As String In OUs.Items
      OUListView.Items.Add(Item)
    Next
    OUListView.Items(0).Selected = True
  End Sub

  Private Sub cmdSelect_OU_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdSelect_OU.Click
    Dim SelectedOU As String =
OUListView.Items(OUListView.SelectedIndex).Value.ToString

    If Not String.Equals(Session("OU"), SelectedOU) Then
      Dim OU As New Message
      OU.Instruction = "Set Directory Level"
      OU.Data = SelectedOU
      Formatter.Serialize(NS, OU)
      Session("OU") = SelectedOU
      Session("UserList_Changed") = "True"
      Session("DisabledUserListed") = "False"
      Session("EnabledUserListed") = "False"
      Session("GroupList_Changed") = "True"
      Session("ClientInfo") = "False"
      Session("Computers_Listed") = "False"
    End If

    If String.Equals(Session("Function"), "Users") Then
```



```
      ActiveForm = frmUserManagement
    ElseIf String.Equals(Session("Function"), "Groups") Then
      ActiveForm = frmGroupManagement
    Else
      ActiveForm = frmClientList
    End If
  End Sub

  Private Sub frmUserList_Activate(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles frmUserList.Activate
    If String.Equals(Session("Function"), "Enable Users") Then

      UserSelectionList_Set(Disabled_Users)

    ElseIf String.Equals(Session("Function"), "Disable Users") Then

      UserSelectionList_Set(Enabled_Users)

    ElseIf String.Equals(Session("Function"), "Delete Users") _
      Or String.Equals(Session("Function"), "Add To Groups_Users") _
      Or String.Equals(Session("Function"), "Remove From
Groups_Users") _
      Or String.Equals(Session("Function"), "List
GroupMembership_Users") _
      Or String.Equals(Session("Function"), "Add Users To Groups")
Then

      UserSelectionList_Set(Users)

    ElseIf String.Equals(Session("Function"), "List Users") Then

      UserListView_set(Users)

    ElseIf String.Equals(Session("Function"), "List Users In Groups")
Then

      UserListView_set(Group_Members)

    ElseIf String.Equals(Session("Function"), "Remove Users From
Groups") Then

      UserSelectionList_Set(Group_Members)
    End If
```

```
    End Sub

    Private Sub cmdUser_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdUser.Click
        Session("Function") = "Users"
        ActiveForm = frmOUList
    End Sub

    Private Sub cmdGroup_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdGroup.Click
        ActiveForm = frmOUList
        Session("Function") = "Groups"
    End Sub

    Private Sub cmdListUser_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles cmdListUser.Click
        Session("Function") = "List Users"

        cmdSelect_User.Visible = False

        UserListView.Visible = True
        UserSelectionList.Visible = False

        getUserList()
    End Sub

    Public Sub getUserList()
        Dim Command As New Message
        If String.Equals(Session("UserList_Changed"), "True") Then

            Command.Instruction = "List Users"
            Formatter.Serialize(NS, Command)

            If Wait4Data() Then
                Request_TimedOut()
                Exit Sub
            End If

            Users = DirectCast(Formatter.Deserialize(NS), ObjectList)
            Session("UserList_Changed") = "False"
        End If

        If Users.Items(0).EndsWith("Found!") Then
```



```
            imgError.Visible = True
            lblErrorTitle.Visible = False
            lblError.Text = Users.Items(0)
            ActiveForm = frmError
        Else
            ActiveForm = frmUserList
        End If
    End Sub

    Sub UserSelectionList_Set(ByVal List As ObjectList)
        UserSelectionList.Items.Clear()
        For Each Item As String In List.Items
            UserSelectionList.Items.Add(Item)
        Next
        UserSelectionList.SelectedIndex = 0
    End Sub

    Sub UserListView_set(ByVal list As ObjectList)
        UserListView.Items.Clear()
        For Each Item As String In list.Items
            UserListView.Items.Add(Item)
        Next
    End Sub

    Private Sub cmdBack_frmUsermanagement_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdBack_frmUserManagement.Click
        ActiveForm = frmMainMenu
    End Sub

    Private Sub cmdCreateUser_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdCreateUser.Click
        Session("Function") = "Add Users"
        ActiveForm = frmAddUser
    End Sub

    Private Sub cmdAddUser_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles cmdAddUser.Click
        If Not txtUser.Text.Equals("") Then
            Dim Command As New Message
            Command.Instruction = "Add Users"
            Command.Data = txtUser.Text
            If OptionSelection.Items(0).Selected = True Then
```



```
                Command.Data = Command.Data + "," + "Yes"
            Else
                Command.Data = Command.Data + "," + "No"
            End If
            If OptionSelection.Items(1).Selected = True Then
                Command.Data = Command.Data + "," + "Yes"
            Else
                Command.Data = Command.Data + "," + "No"
            End If

            Formatter.Serialize(NS, Command)

            If Wait4Data() Then
                Request_TimedOut()
                Exit Sub
            End If

            Command = DirectCast(Formatter.Deserialize(NS), Message)

            imgError.Visible = True
            lblErrorTitle.Visible = False

            If Not Command.Instruction.Equals("Error") Then
                Session("UserList_Changed") = "True"
                lblErrorTitle.Text = Command.Instruction
                imgError.Visible = False
                lblErrorTitle.Visible = True
            End If
            lblError.Text = Command.Data
            ActiveForm = frmError
        End If
    End Sub

    Private Sub cmdBack_frmAdduser_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdBack_frmAddUser.Click
        ActiveForm = frmUserManagement
    End Sub

    Private Sub cmdBack_frmError_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdBack_frmError.Click
        Dim Functions As String = Session("Function")
        If Functions.EndsWith("Users") Then
```



```
            ActiveForm = frmUserManagement
        ElseIf Functions.EndsWith("Groups") Then
            ActiveForm = frmGroupManagement
        ElseIf Functions.EndsWith("Services") Then
            ActiveForm = frmServiceManagement
        ElseIf Functions.EndsWith("Log") Then
            ActiveForm = frmEventViewer
        ElseIf Functions.EndsWith("Server") Then
            ActiveForm = frmMainMenu
        ElseIf lblError.Text.Equals("Error") _
            Or Functions.Equals("Message Client") Then
            ActiveForm = frmClientManagement
        Else
            ActiveForm = frmMainMenu
        End If
    End Sub

    Private Sub cmdEnableUser_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdEnableUser.Click
        Dim Command As New Message
        Session("Function") = "Enable Users"

        cmdSelect_User.Visible = True
        cmdSelect_User.Text = "Enable"

        UserListView.Visible = False
        UserSelectionList.Visible = True
        UserSelectionList.SelectType = MobileControls.ListSelectType.Radio

        If String.Equals(Session("DisabledUserListed"), "False") _
            Or String.Equals(Session("UserList_Changed"), "True") Then

            Command.Instruction = "List Disabled Users"
            Formatter.Serialize(NS, Command)

            If Wait4Data() Then
                Request_TimedOut()
                Exit Sub
            End If

            Disabled_Users = DirectCast(Formatter.Deserialize(NS),
ObjectList)
            Session("DisabledUserListed") = "True"
```

```
    End If
    If Disabled_Users.Items(0).EndsWith("Found!") Then
      imgError.Visible = True
      lblErrorTitle.Visible = False
      lblError.Text = Disabled_Users.Items(0)
      ActiveForm = frmError
    Else
      ActiveForm = frmUserList
    End If
  End Sub

  Private Sub cmdDeleteUser_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdDeleteUser.Click
    Session("Function") = "Delete Users"

    cmdSelect_User.Visible = True
    cmdSelect_User.Text = "Delete"

    UserListView.Visible = False
    UserSelectionList.Visible = True
    UserSelectionList.SelectType = MobileControls.ListSelectType.Radio

    getUserList()
  End Sub

    Private Sub cmdAddUsersToGroup_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdAddUsersToGroup.Click
    Session("Function") = "Add Users To Groups"

    cmdSelect_User.Visible = True
    cmdSelect_User.Text = "Add Users"
    cmdSelect_Group.Visible = True
    cmdSelect_Group.Text = "Select"

    UserSelectionList.Visible = True
    UserListView.Visible = False
    UserSelectionList.SelectType =
MobileControls.ListSelectType.CheckBox

    GroupListView.Visible = False
    GroupSelectionList.Visible = True
```



```
    GroupSelectionList.SelectType =
MobileControls.ListSelectType.Radio

    getGroupList()
  End Sub

  Private Sub cmdSelect_Group_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdSelect_Group.Click
    Dim Functions As String = Session("Function")
    Dim Command As New Message

    If Functions.Equals("List Users In Groups") _
      Or Functions.Equals("Remove Users From Groups") Then

      Session("Group") =
GroupSelectionList.Items(GroupSelectionList.SelectedIndex).ToString
      Command.Instruction = "List Users In Groups"
      Command.Data = Session("Group")
      Formatter.Serialize(NS, Command)

      If Wait4Data() Then
        Request_TimedOut()
        Exit Sub
      End If

      Group_Members = DirectCast(Formatter.Deserialize(NS),
ObjectList)

      If Group_Members.Items(0).EndsWith("Found!") Then
        imgError.Visible = False
        lblErrorTitle.Visible = True
        lblError.Text = Group_Members.Items(0)
        ActiveForm = frmError
      Else
        ActiveForm = frmUserList
      End If

    ElseIf Functions.Equals("Add Users To Groups") Then

      getUserList()
      Session("Group") =
GroupSelectionList.Items(GroupSelectionList.SelectedIndex).ToString
```



```
    ElseIf Functions.Equals("Delete Groups") Then

      Session("Group") =
GroupSelectionList.Items(GroupSelectionList.SelectedIndex).ToString
      Command.Instruction = Functions
      Command.Data = Session("Group")
      Formatter.Serialize(NS, Command)

      If Wait4Data() Then
        Request_TimedOut()
        Exit Sub
      End If

      Command = DirectCast(Formatter.Deserialize(NS), Message)

      imgError.Visible = True
      lblErrorTitle.Visible = False

      If Not Command.Instruction.Equals("Error") Then
        Session("GroupList_Changed") = "True"
        imgError.Visible = False
        lblErrorTitle.Visible = True
        lblErrorTitle.Text = Command.Instruction
      End If

      lblError.Text = Command.Data
      ActiveForm = frmError
    Else
      Dim i As Integer
      Dim User As String = Session("User")

      Command.Instruction = Functions
      For i = 0 To GroupSelectionList.Items.Count - 1
        If GroupSelectionList.Items(i).Selected = True Then
          Command.Data = GroupSelectionList.Items(i).ToString + "," +
Command.Data
        End If
      Next
      Command.Data = Command.Data + "," + User

      If Command.Data.Length > (User.Length + 1) Then

        Formatter.Serialize(NS, Command)
```



```
        If Wait4Data() Then
          Request_TimedOut()
          Exit Sub
        End If

        Command = DirectCast(Formatter.Deserialize(NS), Message)

        imgError.Visible = False
        lblErrorTitle.Visible = True
        lblErrorTitle.Text = Command.Instruction
        lblError.Text = Command.Data
      Else
        imgError.Visible = True
        lblErrorTitle.Visible = False
        lblError.Text = "No Users were selected!"
      End If
      ActiveForm = frmError
    End If
  End Sub


  Private Sub cmdServerLogoff_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdServerLogoff.Click
    Session("Function") = "LogOff Server"
    txtQuestion.Text = "Are you sure you want to logoff the system?"
    ActiveForm = frmConfirmation
  End Sub

  Private Sub cmdServerRestart_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdServerRestart.Click
    Session("Function") = "Restart Server"
    txtQuestion.Text = "Are you sure you want to restart the system?"
    ActiveForm = frmConfirmation
  End Sub

  Private Sub cmdServerShutdown_Click(ByVal sender As Object, ByVal
e As System.EventArgs) Handles cmdServerShutdown.Click
    Session("Function") = "Shutdown Server"
    txtQuestion.Text = "Are you sure you want to shutdown the system?"
    ActiveForm = frmConfirmation
  End Sub
```

```
    Private Sub cmdViewDescription_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdViewDescription.Click
        Dim i As Integer
        Session("Function") = "Description Of Services"
        getServiceList()

        cmdSelect_Service.Visible = True
        cmdSelect_Service.Text = "View Description"
        ServiceSelectionList.Visible = True
        ServiceListView.Visible = False

        ServiceSelectionList.Items.Clear()
        For i = 0 To 19
            If String.Equals(Session("Managing"), "Server") Then

ServiceSelectionList.Items.Add(Server_Services.DisplayName(i))
            Else
                ServiceSelectionList.Items.Add(Client_Services.DisplayName(i))
            End If
        Next
        ServiceSelectionList.Items(0).Selected = True

        ActiveForm = frmServiceList
    End Sub

    Private Sub Command4_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Command4.Click
        ActiveForm = frmServiceManagement
    End Sub

    Private Sub KeepAlive_Elapsed(ByVal sender As System.Object, ByVal
e As System.Timers.ElapsedEventArgs) Handles KeepAlive.Elapsed
        Dim Command As New Message
        Command.Instruction = "Test"
        Formatter.Serialize(NS, Command)
    End Sub

    Private Sub cmdBack_frmUserList_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cmdBack_frmUserList.Click
        Dim Functions As String = Session("Function")
        If Functions.EndsWith("Users") Then
```



```
            ActiveForm = frmUserManagement
        ElseIf Functions.EndsWith("Groups") Then
            ActiveForm = frmGroupManagement
        End If
    End Sub

    Sub getEventLogs()
        If String.Equals(Session("EventLogListed"), "False") Then

            Dim Command As New Message
            Command.Instruction = "Get EventLog"
            Command.Data = Session("Managing") & "," & Format(EventDate,
"dd/MM/yyyy")
            Formatter.Serialize(NS, Command)
            Session("EventLogListed") = "True"
        End If
    End Sub

    Sub List_Event()
        lblType.Text = "Type    : " & Entries(Position).EntryType.ToString
        lblTime.Text    =    "Time                    :    "    &
Entries(Position).TimeWritten.TimeOfDay.ToString
        lblUser.Text = "User    : " & Entries(Position).UserName
        lblComputer.Text = "Computer : " & Entries(Position).MachineName
        lblEventID.Text = "EventID  : " & Entries(Position).EventID
        txtMessage.Text = Entries(Position).Message

        If Position = 0 Then
            cmdPrevious.Visible = False
        Else
            cmdPrevious.Visible = True
        End If

        If Position = Entries.Length - 1 Then
            cmdNext.Visible = False
        Else
            cmdNext.Visible = True
        End If
        ActiveForm = frmEventDescription
    End Sub

    Private Sub cmdApplication_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdApplication.Click
```



```
        getEventLogs()
        Session("Function") = "Application Log"

        If String.Equals(Session("Managing"), "Server") Then
            If EventLog_Server.App_Entries_Present = True Then
                Entries = EventLog_Server.Application
                imgEventLog.ImageUrl = "images/Application.gif"
                lblDate1.Text = "( " & EventDate & " )"
                Position = 0
                List_Event()
            Else
                imgError.Visible = True
                lblErrorTitle.Visible = False
                lblError.Text = "No log entries found in the server!"
                ActiveForm = frmError
            End If
        Else
            If EventLog_Client.App_Entries_Present = True Then
                Entries = EventLog_Client.Application
                imgEventLog.ImageUrl = "images/Application.gif"
                lblDate1.Text = "( " & EventDate & " )"
                Position = 0
                List_Event()
            Else
                imgError.Visible = True
                lblErrorTitle.Visible = False
                lblError.Text = "No log entries found in the client!"
                ActiveForm = frmError
            End If
        End If
    End Sub

    Private Sub cmdSystem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdSystem.Click
        getEventLogs()
        Session("Function") = "System Log"

        If String.Equals(Session("Managing"), "Server") Then
            If EventLog_Server.Sys_Entries_Present = True Then
                Entries = EventLog_Server.System
                imgEventLog.ImageUrl = "images/System.gif"
                lblDate1.Text = "( " & EventDate & " )"
                Position = 0
```



```
                List_Event()
            Else
                imgError.Visible = True
                lblErrorTitle.Visible = False
                lblError.Text = "No log entries found in the server!"
                ActiveForm = frmError
            End If
        Else
            If EventLog_Client.Sys_Entries_Present = True Then
                Entries = EventLog_Client.System
                imgEventLog.ImageUrl = "images/System.gif"
                lblDate1.Text = "( " & EventDate & " )"
                Position = 0
                List_Event()
            Else
                imgError.Visible = True
                lblErrorTitle.Visible = False
                lblError.Text = "No log entries found in the client!"
                ActiveForm = frmError
            End If
        End If
    End Sub

    Private Sub cmdDirectory_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdDirectory.Click
        getEventLogs()
        Session("Function") = "Directory Log"

        If EventLog_Server.Dir_Entries_Present = True Then
            Entries = EventLog_Server.Directory_Service
            imgEventLog.ImageUrl = "images/Directory.gif"
            lblDate1.Text = "( " & EventDate & " )"
            Position = 0
            List_Event()
        Else
            imgError.Visible = True
            lblErrorTitle.Visible = False
            lblError.Text = "No log entries found!"
            ActiveForm = frmError
        End If
    End Sub
```

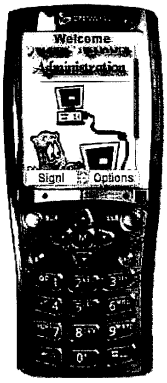
```
    Private Sub cmdDNSServer_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdDNSServer.Click
        getEventLogs()
        Session("Function") = "DNS Log"

        If EventLog_Server.DNS_Entries_Present = True Then
            Entries = EventLog_Server.DNS_Server
            imgEventLog.ImageUrl = "images/DNS.gif"
            lblDate1.Text = "( " & EventDate & " )"
            Position = 0
            List_Event()
        Else
            imgError.Visible = True
            lblErrorTitle.Visible = False
            lblError.Text = "No log entries found!"
            ActiveForm = frmError
        End If
    End Sub
    Private Sub cmdSend_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdSend.Click
        Dim Command As New Message
        Command.Instruction = "Popup Message"
        If Not Trim(txtPopupMessage.Text).Equals("") Then
            Command.Data = txtPopupMessage.Text
            If String.Equals(Session("Function"), "Message Server") Then
                If ChoiceSelectionList.Items(0).Selected = True Then
                    Command.Data = Command.Data + "|" + "Server"
                Else
                    Command.Data = Command.Data + "|" + "Broadcast"
                End If
            Else
                Command.Data = Command.Data + "|" + Session("Managing")
            End If
            Formatter.Serialize(NS, Command)

            ActiveForm = frmError
            imgError.Visible = False
            lblErrorTitle.Visible = True
            lblErrorTitle.Text = "Succesfully Send"
            lblError.Text = "Message was successfully send!"
        End If
    End Sub
```


9.2 SAMPLE OUTPUT

WELCOME SCREEN



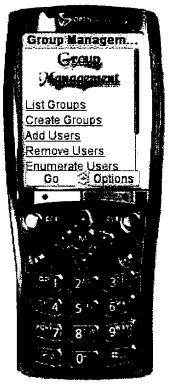
LOGIN SCREEN

AUTHENTICATION SUCCESSFUL

AUTHENTICATION FAILURE

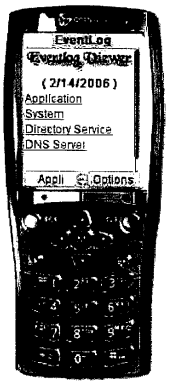
FUNCTIONS DISPLAY SCREEN

GROUP MANAGEMENT SCREEN



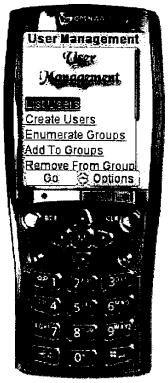
EVENT LOG DATE SELECTION

EVENT LOG VIEWER



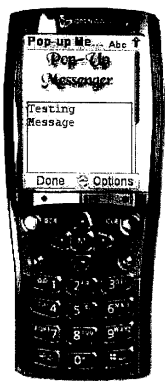
DOMAIN SELECTION SCREEN

USER MANAGEMENT SCREEN

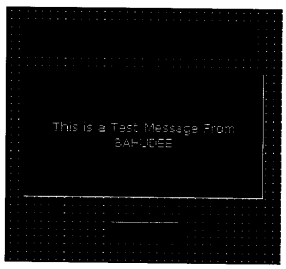


SERVICE MANAGEMENT SCREEN

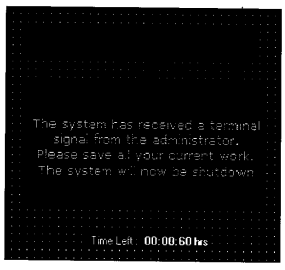
MESSAGE SENDING SCREEN



CLIENT SIDE MESSAGE DELIVERY SCREEN



CLIENT SYSTEM ABORTING SCREEN



10. REFERENCES

1. R.E.Woods,"Microsoft Windows 2000 Active Directory Services", Prentice-Hall of India Private Limited, 2000.

2. Andy Wigley and Peter Roxburgh (2002), "Building .NET Applications For Mobile Devices", WP Publishers & Distributors (P) Ltd.

3. Roger S. Pressman (2001), "Software Engineering a Practitioner's Approach", McGraw Hill.

4. Bill Evjen, Billy Hollis, Rockford Lhotka, , Rama Ramachandran, Bill Sheldon ,"Professional VB.NET 2003", WP Publishers & Distributors (P) Ltd.

5. Tim McCarthy, Jonathan Pinnock , "Programming ASP.NET", Prentice-Hall of India Private Limited.