**CHAPTER 1**

**INTRODUCTION**

Most of the organization they connect more numbers of systems to form a network to make their work easier to share their files and folders. While connecting we want to monitor the network system activities for secure purpose. This Project deals with monitoring the Network Screen Activities. It has two methodologies one for Client and another for Server. In the proposed system we introduce current session option to monitor the network systems at the same time and in the accesses folder option shows the username and user accessed folders. While client logins to the server, the Client IP Address and System names are added to the server. Server will display all the user names, from server we can monitor the particular Client Screen Activities like that currently opened screens, what are all the files created, modified and deleted.

Remote Desktop Services is one of Microsoft Windows components to access a remote computer through the network. Only the user interface of the application is presented at the client. Any input is redirected over to the remote computer over the network. At work, we use Remote Desktop a great deal. It allows us to login to a remote server to perform health checks, deploy applications, troubleshoot problems, etc. We also use remote desktop often when we do WFH (work from home).

This project is an effort in to develop a simple IP Subnet Calculator tool only for class C IP. The IP Subnet Calculator was to give the user a quick and interactive method to calculate available subnet and hosts. Due to the repetitiveness of such calculates, tools such as an IP Subnet Calculator were developed to eliminate common mathematical mistakes. Furthermore, these tools also provide a means for the user to do such calculations without actually understanding the details behind calculating IP subnets.

You have several choices for your email client application, including a range of web email services. However, many people prefer to use a stand-alone application designed specifically for email since they sometimes have a better focus and features. Some people avoid the Microsoft Outlook application because of its historic vulnerability to viruses.

* 1. **Objection of the project**

In Order to be able to define our system architecture, we must first dearly state what our objective that will driver system behavior at the same one of our objective is to create an experience, which is not only unique to the (user) client, but also makes him feel that he has loyal attachment to the system and approaches us whenever heshe needs.

To achieve better results and success by implement computerize

ed process instead of manual process.

**1.2 Module and their Description**

This project having Five Modules

1. Admin
2. Network Monitoring System
3. Remote Desktop Monitoring
4. E-Mail Client
5. IP Calculator

1.Admin

The Login module consists of username and password. This process is for authenisation.The

user name and password is correct it is link into next page. This process is done in login.

2. Network Monitoring System:

The most of the organization they connect more numbers of systems to form a network to make their work easier to share their files and folders. While connecting we want to monitor the network system activities for secure purpose. This Project deals with monitoring the Network Screen Activities. It has two methodologies one for Client and another for Server. In the proposed system we introduce current session option to monitor the network systems at the same time and in the accesses folder option shows the username and user accessed folders. While client logins to the server, the Client IP Address and System names are added to the server. Server will display all the user names, from server we can monitor the particular Client Screen Activities like that currently opened screens, what are all the files created, modified and deleted

3. Remote Desktop Monitoring

Remote Desktop Services is one of Microsoft Windows components to access a remote computer through the network. Only the user interface of the application is presented at the client. Any input is redirected over to the remote computer over the network. At work, we use Remote Desktop a great deal. It allows us to login to a remote server to perform health checks, deploy applications, troubleshoot problems, etc. We also use remote desktop often when we do WFH (work from home).

**4. E-Mail Client**

You have several choices for your email client application, including a range of web email services. However, many people prefer to use a stand-alone application designed specifically for email since they sometimes have a better focus and features. Some people avoid the Microsoft Outlook application because of its historic vulnerability to viruses. An application that runs on a personal computer or workstation and enables you to send, receive and organize e-mail. It's called a client because e-mail systems are based on a client-server architecture. Mail is sent from many clients to a central server, which re-routes the mail to its intended destination.

**5. IP CALCULATOR SUBNET**

The calculator module certain process

* **Internet protocol**
* **Network type**
* **Calculate**

The calculate is used to calculate the network type and it class. The other process are prefix,net.mas,maximum number of host,network,broadcast,iprange,maximum number of subnets, wanted number of hosts.

**Internet protocol:**

The Internet protocol is used to know about the network type. The protocol is enter in the textbox and calculates. It will show the network type. If wrong protocol is entering in the textbox. It will show the error message.

**Network type:**

The Network used to describes the class.

* + - **Class A:0: 127 255**
    - **Class B:128:191 255**
    - **ClassC:192:223 255**
    - **ClassD:224:239 255**
    - **ClassE:239:254 255**

**Calculate:**

The Calculate is used to calculate the internet protocol and network type.In this system Maximum number of subnet is display.

**CHAPTER 3**

CUSTOMER REQUIREMENTS DETERMINATION

**3.1Existing system**

The existing system has been maintained manually. The system, which has been maintained manually, had been complex and complicated. There were many chances to loss the data and the work wouldn’t be an effective and efficient one. Manual operation is always been complicated to the organizations for maintaining the records. In the existing system

* In this system we can’t monitor who all are accessing the files at the same time.
* It’s difficult to find which are the files are updated and renamed.

**Drawbacks of the existing system**

* Maintenance of the system is very difficult.
* Employee’s attentions are needed for maintaining the system.
* There is a possibility for getting inaccurate results.
* User friendliness is very less.
* It consumes more time for processing the activities.

**3.2Proposed system**

* Considering the anomalies in the existing system computerization of the whole activity is being suggested after initial analysis by the organization.
* In this system we can monitor who all are accessing the files in the network at the same time.
* When monitoring the connected systems, we can easily found what are the files updated and renamed.

**ADVANTAGES OF THE PROPOSED SYSTEM**

* Fully Secured
* Role based access
* Ease in maintenance
* Notification about the modification

**CHAPTER 4**

**SOFTWARE REQUIREMENTS SPECIFICATION**

Software Requirements Specification (SRS) is the starting point of the software development activity. Little importance was given to this phases in the early days of software development. The emphasis was first on coding and then shifted to design.

As systems grew more complex, it become evident that the goal of the entire system cannot be easily comprehended. Hence need for the requirements analysis phase arose. Now, for large software systems, requirements analysis is perhaps the most difficult activity and also the most error prone.

Some of the difficulty is due to the scope of this phase. The software project is imitated by the client needs. In the beginning these needs are in the minds of various people in the client organization. The requirement analyst has to identify the requirements by tacking to these people and understanding their needs. In situations where the software is to automated a currently manuals process, most of the needs can be understood by observing the current practice.

The SRS is a means of translating the ideas in the minds of the clients (the output) into formal document (the output of the requirements phase). Thus the output of the phase is a set of formally specified requirements, which hopefully are complete and consistent, while the input has none of these properties.

* 1. **Functional Requirements**

**Start**

**Evaluation**

**Subnet Mask Calculator**

**Sending Email**

**Attached Email**

**Disconnect**

**Monitor Remote Desktop**

**Connect FTP server**

**Stop Monitoring**

**Getting Current session**

**Start Monitoring**

**IP Address**

**4.2 Performance Requirements**

The project must the end user requirements. Accuracy and fast must be imposed in the Project.

The project is development as easy as possible for the sake of end user. The project has to be developed with view of satisfying the future requirements and future enhancement.

The tool has been finally implemented satisfying the needs specified by the company. As per the performance is concerned this system said is performing

This processing as well as tine taken to generate well reports where also even when large amount of data was used. The system is designed in such a way that even when large amount of data used for processing there would less performance degradation.

**4.3 Interface requirements**

**4.3.1 Hardware Interface**

The stranded input device like keyboard and mouse are to get input. The output will be generated and display in the monitor. The reports can also be exported to a SQL-server document are text file. The stranded printer in used to take outputs.

**4.3.2 Software Interface**

The design part and interface id done the front end ASP.Net and SQL server as a backend of the project.

**4.4 Operational requirements**

The database or databases that are being failed over to the stand by server cannot be used for anything else. But databases on the standby server not being used for failover can still be used normally.

When it comes time for actual failover, you much one of two things to make your application work either rename the standby server the same name as the failed production server(and the IP address),or re-point your user’s applications to new standby server in some cases, neither of this option is practical.

**4.5 Resource Requirements**

**4.5.1 Hardware Requirements**

PROCESSOR : PENTIUM III 866 MHz

RAM : 128 MD SD RAM

MONITOR : 15” COLOR

HARD DISK : 20 GB

FLOPPY DRIVE : 1.44 MB

CD DRIVE : LG 52X

KEYBOARD : STANDARD 102 KEYS

MOUSE : 3 BUTTONS

**4.5.2 Software Requirements**

OPERATING SYSTEM : Windows XP Professional

ENVIRONMENT : Visual Studio .NET 2008

.NET FRAMEWORK : Version 3.5

LANGUAGE : C#.NET

WEB TECHNOLOGY : ASP.NET

BACKEND : SQL SERVER 2005

**4.6 Security Requirements**

Web application are available via network access, it is a difficult. If not possible, to limit the population of the end-user who may access the applications? In order to product sensitive connect and provide secure mode be implemented throughout the infrastructure that the supports web application and within the application itself.

Web Application have become heavy integrated with critical corporate and database.

E-commerce application extracts and then store sensitive customer information.

**4.7 Design Requirements**

To create project, add base masters and masters to the project, assign behaviors to the master, create and assign behavior sets, and then apply, test and validate those behaviors. It also shows how to create and build a stencil to hold the shapes.

**4.8 Quality and Reliability Requirements**

A software component that is developed for reuse would be correct and would contain no defects. In reality, formal verification is not carried out routinely, and defects can add to occur.However,with each reuse,,defects are found eliminated, and a components qualify improve as a result. Over time the components virtually defect free.

Software reliability is defined in statical term as” the probability of faultier-free operation of a computer program in a specified environment for specified tine”. The software quality and reliability, failure is nonconformance to software requirements. Failure can be only anything or catastrophic. one failure can be corrected within seconds while another requirements week even mouths to correct. Complicating the issue even further, the correction of the one failure may in fact result in the introduction of the errors that ultimately result in other failure.

Web Correct link processing

Application Reliability Error recovery

Quality Input validation and recovery

**CHAPTER 5**

**SYSTEM ANALYSIS**

In this section discussed about data flow diagram, Entity relationship diagram. these things are represented as diagrams with proper notation.

**5.1 Data Flow Diagram**

The data flow diagram is one of the most improvement tools used by the system analyst DeMacro (1978) Nad Gand Sarson (1979) popularized the use if the data flow diagram as modeling tools through their structured system analysis methodologies.

A data flow diagram should be the first tool used by system analyst to model system components. These components are the system processes; the data used by this processes and external entities that interact with the system and the information flows in the system.

There are four kinds of system components

**5.1.1. Process**

Process show what system does. Each process has one or more data inputs and produce one or more data output, Circles in a data flow diagram represent process. Each process has unique name and number. This name and number appear inside the circle that represents the processes in a data flow diagram.

This process is represented as circle

**5.1.2. Data Stores**

File or data store is depositary of data. They contain data that is retained in the system. Processes can enter the data into a data store or retrieve data from the data store. Each data store is represented by thin line in the data flow diagram and each data store has a unique name.

The data store is represented in form of a line

**5.1.3 External Entities**

External entities are outside the system but they either supply input data into the system or use the system output, they are entities which the designer has no control. Square or rectangle may represent external entities that supply data into a system or some times called sources. External entities that use the system data are sometimes called sinks.

**5.1.4 Data Flows**

Dataflow model the passage of data in the system and are represented lines joining system components. An arrow indicates the direction of the flow and the line labeled by the name of the data flow.

**DFD-1**

IP Address

Stop Monitoring

Start Monitoring

Folder Watcher

Share Folder

Accessed Folder

Current Session

**DFD2**

IP Subnet Calculator

Internet Protocol

Calculation

Network Type

**DFD-3**

Remote Desktop

Disconnect

Connect

Authentication

Server Name

**5.2 ER DIAGRAM**

**Start**

**Evaluation**

**Subnet Mask Calculator**

**Sending Email**

**Attached Email**

**Disconnect**

**Monitor Remote Desktop**

**Connect FTP server**

**Start Monitoring**

**Getting Current session**

**Stop Monitoring**

**IP Address**

* 1. **Use Case Diagram**

**Server**

Client

**CHAPTER 6**

**SYSTEM DESIGN**

Design is multi-step process that focuses on data structure software architecture, procedural details, (algorithms etc.) and interface between modules. The design process also translates the requirements into the presentation of software that can be accessed for quality before coding begins.

Computer software design changes continuously as new methods; better analysis and broader understanding evolved. Software Design is at relatively early stage in its revolution.

Therefore, Software Design methodology lacks the depth, flexibility and quantitative nature that are normally associated with more classical engineering disciplines. However techniques for software designs do exist, criteria for design qualities are available and design notation can be applied.

6.1 INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer-based format. Input design is one of the most expensive phases of the operation of computerized system and is often the major problem of a system.

In the project, the input design is made in various web forms with various methods. For example, in the Admin form, the empty username and password is not allowed. The username if exists in the database, the input is considered to be invalid and is not accepted.

6.2 OUTPUT DESIGN

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

In the project, if the employee has to communicate with other employees they can communicate through send and receive message.

**6.3 INTERFACE DESIGN**

The ODBC (Open Database Connectivity) interface is a pure .NET to execute SQL statement. The ODBC provides a set classes and interfaces that can be used by developers to write database applications. Basic ODBC interactions in its simplest form, can be broken down into four steps:

1. Open a connection to the database.

2. Execute a SQL statement

3. Process the result

4. Close the connection to the database

**6.4 TABLE AND DATABASE DESIGN:**

**6.4.1 Normalization:**

Normalization is the process of strutting relational database schema such that most ambiguity is removed. The stage of normalization are referred to as forms and progress from the least restrictive(first normal form)through the most restrictive(Fifth normal form), generally , most database designers do not attempt to implement anything higher then normal form of Boyce code Normal Form.

**6.4.1.1FIRST NORMAL FORM**:

A relation is said to be in First normal form (INF) if and each attributed of the relation is atomic. More simply, to be INF, each column must contain only a single value and each now contain in the same column.

**6.4.1.2 SECOND NORMAL FORM:**

In the Second normal Form, a relation must first fulfill the requirement to be in first Normal Form. Additional, each donkey attribute in the relation must be functionality dependent upon the primary key**.**

**6.4.1.3 THIRD NORMAL FORM:**

A table is said to be in third normal form and every non key attribute is functionality dependent only on the primary key. This normalization process is applied to this system and the normalized tables are given in the above section.

**TABLE DESIGN**

**Admin Login Table**

****

**6.4.2 Database Design:**

The database design is a must for any application developed especially more for the data store projects. Since the chatting method involves storing the message in the table and produced to the sender and receiver, proper handling of the table is a must.

In the project, login table is designed to be unique in accepting the username and the length of the username and password should be greater than zero

The complete listing of the tables and their fields are provided in the annexure under the title ‘Table Structure’.

**6.5 FRONT END DESIGN**

**FEATURES OF ASP.NET**

ASP.NET is the next version of Active Server Pages (ASP); it is a unified Web development platform that provides the services necessary for developers to build enterprise-class Web applications. While ASP.NET is largely syntax compatible, it also provides a new programming model and infrastructure for more secure, scalable, and stable applications.

ASP.NET is a compiled, NET-based environment, we can author applications in any .NET compatible language, including Visual Basic .NET, C#, and JScript .NET. Additionally, the entire .NET Framework is available to any ASP.NET application. Developers can easily access the benefits of these technologies, which include the managed common language runtime environment (CLR), type safety, inheritance, and so on.

ASP.NET has been designed to work seamlessly with WYSIWYG HTML editors and other programming tools, including Microsoft Visual Studio .NET. Not only does this make Web development easier, but it also provides all the benefits that these tools have to offer, including a GUI that developers can use to drop server controls onto a Web page and fully integrated debugging support.

Developers can choose from the following two features when creating an ASP.NET application.Web Forms and Web services, or combine these in any way they see fit. Each is supported by the same infrastructure that allows you to use authentication schemes, cache frequently used data, or customize your application's configuration, to name only a few possibilities.

Web Forms allows us to build powerful forms-based Web pages. When building these pages, we can use ASP.NET server controls to create common UI elements, and program them for common tasks. These controls allow we to rapidly build a Web Form out of reusable built-in or custom components, simplifying the code of a page.

An XML Web service provides the means to access server functionality remotely. Using Web services, businesses can expose programmatic interfaces to their data or business logic, which in turn can be obtained and manipulated by client and server applications. XML Web services enable the exchange of data in client-server or server-server scenarios, using standards like HTTP and XML messaging to move data across firewalls. XML Web services are not tied to a particular component technology or object-calling convention. As a result, programs written in any language, using any component model, and running on any operating system can access XML Web services

Each of these models can take full advantage of all ASP.NET features, as well as the power of the .NET Framework and .NET Framework common language runtime. Accessing databases from ASP.NET applications is an often-used technique for displaying data to Web site visitors. ASP.NET makes it easier than ever to access databases for this purpose. It also allows us to manage the database from your code .

ASP.NET provides a simple model that enables Web developers to write logic that runs at the application level. Developers can write this code in the global.aspx text file or in a compiled class deployed as an assembly. This logic can include application-level events, but developers can easily extend this model to suit the needs of their Web application.

ASP.NET provides easy-to-use application and session-state facilities that are familiar to ASP developers and are readily compatible with all other .NET Framework APIs.ASP.NET offers the IHttpHandler and IHttpModule interfaces. Implementing the IHttpHandler interface gives you a means of interacting with the low-level request and response services of the IIS Web server and provides functionality much like ISAPI extensions, but with a simpler programming model. Implementing the IHttpModule interface allows you to include custom events that participate in every request made to your application.

ASP.NET takes advantage of performance enhancements found in the .NET Framework and common language runtime. Additionally, it has been designed to offer significant performance improvements over ASP and other Web development platforms. All ASP.NET code is compiled, rather than interpreted, which allows early binding, strong typing, and just-in-time (JIT) compilation to native code, to name only a few of its benefits. ASP.NET is also easily factorable, meaning that developers can remove modules (a session module, for instance) that are not relevant to the application they are developing.

ASP.NET provides extensive caching services (both built-in services and caching APIs). ASP.NET also ships with performance counters that developers and system administrators can monitor to test new applications and gather metrics on existing applications. Writing custom debug statements to your Web page can help immensely in troubleshooting your application's code. However, it can cause embarrassment if it is not removed. The problem is that removing the debug statements from your pages when your application is ready to be ported to a production server can require significant effort.

ASP.NET offers the Trace Context class, which allows us to write custom debug statements to our pages as we develop them. They appear only when you have enabled tracing for a page or entire application. Enabling tracing also appends details about a request to the page, or, if you so specify, to a custom trace viewer that is stored in the root directory of your application. The .NET Framework and ASP.NET provide default authorization and authentication schemes for Web applications. we can easily remove, add to, or replace these schemes, depending upon the needs of our application .

ASP.NET configuration settings are stored in XML-based files, which are human readable and writable. Each of our applications can have a distinct configuration file and we can extend the configuration scheme to suit our requirements.

**DATA ACCESS WITH ADO.NET**

As you develop applications using ADO.NET, you will have different requirements for working with data. You might never need to directly edit an XML file containing data - but it is very useful to understand the data architecture in ADO.NET.

ADO.NET offers several advantages over previous versions of ADO

Interoperability

Maintainability

Programmability

Performance Scalability

**INTEROPERABILITY**

ADO.NET applications can take advantage of the flexibility and broad acceptance of XML. Because XML is the format for transmitting datasets across the network, any component that can read the XML format can process data. The receiving component need not be an ADO.NET component.

The transmitting component can simply transmit the dataset to its destination without regard to how the receiving component is implemented. The destination component might be a Visual Studio application or any other application implemented with any tool whatsoever.

The only requirement is that the receiving component be able to read XML. SO, XML was designed with exactly this kind of interoperability in mind.

**MAINTAINABILITY**

In the life of a deployed system, modest changes are possible, but substantial, Architectural changes are rarely attempted because they are so difficult. As the performance load on a deployed application server grows, system resources can become scarce and response time or throughput can suffer. Faced with this problem, software architects can choose to divide the server's business-logic processing and user-interface processing onto separate tiers on separate machines.

In effect, the application server tier is replaced with two tiers, alleviating the shortage of system resources. If the original application is implemented in ADO.NET using datasets, this transformation is made easier.

ADO.NET data components in Visual Studio encapsulate data access functionality in various ways that help you program more quickly and with fewer mistakes.

**PERFORMANCE**

ADO.NET datasets offer performance advantages over ADO disconnected record sets. In ADO.NET data-type conversion is not necessary.

**SCALABILITY**

ADO.NET accommodates scalability by encouraging programmers to conserve limited resources. Any ADO.NET application employs disconnected access to data; it does not retain database locks or active database connections for long durations.

##### VISUAL STUDIO .NET

Visual Studio .NET is a complete set of development tools for building ASP Web applications, XML Web services, desktop applications, and mobile applications In addition to building high-performing desktop applications, you can use Visual Studio's powerful component-based development tools and other technologies to simplify team-based design, development, and deployment of Enterprise solutions.

Visual Basic .NET, Visual C++ .NET, and Visual C# .NET all use the same integrated development environment (IDE), which allows them to share tools and facilitates in the creation of mixed-language solutions.

In addition, these languages leverage the functionality of the .NET Framework and simplify the development of ASP Web applications and XML Web services. Visual Studio supports the .NET Framework, which provides a common language runtime and unified programming classes; ASP.NET uses these components to create ASP Web applications and XML Web services. Also it includes MSDN Library, which contains all the documentation for these development tools.

##### THE .NET FRAMEWORK

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

**OBJECTIVES OF. NET FRAMEWORK**

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.

2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.

3. Eliminates the performance problems

There are different types of application, such as Windows-based applications and Web-based applications.

To make communication on distributed environment to ensure that code be accessed by the .NET Framework can integrate with any other code.

##### COMPONENTS OF .NET FRAMEWORK

**1. THE COMMON LANGUAGE RUNTIME (CLR)**

The common language runtime is the foundation of the .NET Framework. It manages code at execution time, providing important services such as memory management, thread management, and remoting and also ensures more security and robustness. 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

**THE .NET FRAME WORK CLASS LIBRARY**

It is a comprehensive, object-oriented collection of reusable types used 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.

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 to enables embeds managed components or Windows Forms controls in HTML documents.

**FEATURES OF THE COMMON LANGUAGE RUNTIME**

The common language runtime manages memory; thread execution, code execution, code safety verification, compilation, and other system services these are all run on CLR.

Security.

Robustness.

Productivity.

Performance.

**Security**

The runtime enforces code access security. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally feature rich. With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin to perform file-access operations, registry-access operations, or other sensitive functions.

**ROBUSTNESS**

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system(CTS). The CTS ensures that all managed code is self-describing. The managed environment of the runtime eliminates many common software issues

**PRODUCTIVITY**

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers.

**PERFORMANCE**

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS).

**6.6 BACK END DESIGN**

# FEATURES OF SQL-SERVER 2000

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Service

SQL-SERVER database consist of six type of objects,

They are,

1. TABLE

2. QUERY

3. FORM

4. REPORT

5. MACRO

**TABLE:**

A database is a collection of data about a specific topic.

**VIEWS OF TABLE:**

We can work with a table in two types,

1. Design View

2. Datasheet View

**Design View**

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

**Datasheet View**

To add, edit or analyses the data itself we work in tables datasheet view mode.

**QUERY**

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dataset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dataset. Access either displays the dataset or snapshot for us to view or perform an action on it, such as deleting or updating.

**FORMS**

A form is used to view and edit information in the database record by record .A form displays only the information we want to see in the way we want to see it. Forms use the familiar controls such as textboxes and checkboxes. This makes viewing and entering data easy.

**Views of Form**

We can work with forms in several primarily there are two views,

They are,

**1. Design View**

**2. Form View**

**Design View**

To build or modify the structure of a form, we work in forms design view. We can add control to the form that are bound to fields in a table or query, includes textboxes, option buttons, graphs and pictures.

**Form View**

The form view which display the whole design of the form.

**REPORT**

A report is used to vies and print information from the database. The report can ground records into many levels and compute totals and average by checking values from many records at once. Also the report is attractive and distinctive because we have control over the size and appearance of it.

**MACRO**

A macro is a set of actions. Each action in macros does something. Such as opening a form or printing a report .We write macros to automate the common tasks the work easy and save the time.

**6.7 Algorithm Used**

Step1: The username and password is entered it redirectly to the next page.

Step2: To view the main network page, after it may be share folder to the current session.

Step3: The next page contsins the details of the client of sharing other systems.

Step4: After the form contains the uset details may be view of the current session.

Step5: The next page contains the IP Subnet calculator, you must enter the IP address and then calculate the range of netmask.

Step6: In the above page may contains the some information prefix, netmask, network, broadcast, maximum numbers of subnets.

Step7: In another form contains “Remote desktop, you must enter the Server Name, Username, and Password and then connect or disconnect.

Step8: After the user must access to the server using Remote Desktop.

**CHAPTER 7**

**CODING**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Management;

using System.IO;

using CustomUIControls;

using System.Data.OleDb;

using System.Threading;

namespace NetWorkSpy

{

public partial class Form1 : Form

{

public bool flage=true;

public bool flager = true;

public bool flaget = true;

public bool flagec = true;

public bool Ostate = false;

FileStream fs ;

StreamWriter m\_streamWriter;

StreamWriter m\_streamWriter2;

StreamWriter m\_streamWriter3;

StreamWriter m\_streamWriter4;

string usname = "";

DataTable dt = new DataTable();

DataTable dt1 = new DataTable();

DataTable dt2 = new DataTable();

DataTable dt3 = new DataTable();

ConnectionOptions co;

ManagementScope ms;

ObjectQuery oq;

ObjectQuery oq1;

ObjectQuery oq2;

public int \_RefreshTime = 1;

public string RemoteIPAddress="";

public string RemoteUserName="";

public string RemoteActiveTime="";

public string totalAccessFile="";

public string RemoteOS="";

public bool \_ShowPopUp=true;

public int Timcnt = 0;

public string IDealTime = "";

public int oldcount = 0, newcount = 0;

public string RDirPath = "";

public static string sDBPath = @".\spyDB.mdb";

public static string ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + sDBPath + ";Persist Security Info=False;";

public int cCount = 0;

public int dCount = 0;

public int cHCount = 0;

public int rCount = 0;

public OleDbConnection conn = new OleDbConnection();

FileSystemWatcher[] mywatcher\_List = new FileSystemWatcher[100];

public int iWatcherCount = 0;

public int iSharedCount = 0;

public string[] sFileWatcherPath= new string[100];

TaskbarNotifier taskbarNotifier1 = new TaskbarNotifier();

Timer tim = new Timer();

public int \_CurrentActiveTab;

public string \_CreateLogFilePath="";

public string \_DeleteLogFilePath="";

public string \_RenamedLogFilePath = "";

public string \_ChangedLogFilePath = "";

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

ip.Text = System.Environment.MachineName;

cb.Checked = true;

ListView\_init();

}

public void call()

{

conn.Close();

conn.ConnectionString = ConnectionString;

Auto\_Refresh(); Call AutoRefresh

TaskBar\_Initilization(); Initilize TaskBar Notification

Set The Notify Icon

NotifyIcon.ShowBalloonTip(1);

listView1.Items.Clear();

lviewSession.Items.Clear();

lview.Items.Clear();

Read\_Shared\_Folder(ip.Text, uname.Text, pass.Text, cb.Checked);

Read\_Current\_Session();

\_CreateLogFilePath = Directory.GetCurrentDirectory() + @"\" + "Create\_Log.Txt";

\_DeleteLogFilePath = Directory.GetCurrentDirectory() + @"\" + "Delete\_Log.Txt";

\_RenamedLogFilePath = Directory.GetCurrentDirectory() + @"\" + "Renamed\_Log.Txt";

\_ChangedLogFilePath = Directory.GetCurrentDirectory() + @"\" + "Changed\_Log.Txt";

DataColumn dc = new DataColumn();

}

private void Form1\_Resize(object sender, System.EventArgs e)

{

Hide The Form when it's minimized

if (FormWindowState.Minimized == WindowState)

Hide();

}

private void NotifyIcon\_DoubleClick(object sender, System.EventArgs e)

{

Show the form when Dblclicked on Notifyicon

Show();

WindowState = FormWindowState.Normal;

}

Area For All Menu of System Tray icon ......

Click on Restore...

private void MnuRestore\_Click(object sender, EventArgs e)

{

Show();

WindowState = FormWindowState.Normal;

}

Click on Close...

private void MnuClose\_Click(object sender, EventArgs e)

{

Application.Exit();

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Methods for initilizing List View Header in Shared Folder

private void ListView\_init()

{

List View For Shared Folder........

listView1.View = View.Details;

listView1.Columns.Add("Shared", -1, HorizontalAlignment.Left);

listView1.Columns.Add("Path", -1, HorizontalAlignment.Left);

listView1.Columns.Add("Description", -1, HorizontalAlignment.Left);

listView1.Columns.Add("Status", -1, HorizontalAlignment.Left);

listView1.Columns[0].Width = 100;

listView1.Columns[1].Width = 200;

listView1.Columns[2].Width = 200;

listView1.Columns[3].Width = 75;

List View For Current Session

lviewSession.View = View.Details;

lviewSession.Columns.Add("IP Address", -1, HorizontalAlignment.Left);

lviewSession.Columns.Add("User Name", -1, HorizontalAlignment.Left);

lviewSession.Columns.Add("Access Time(Min)", -1, HorizontalAlignment.Left);

lviewSession.Columns.Add("Ideal Time(Min)", -1, HorizontalAlignment.Left);

lviewSession.Columns.Add("Remote OS", -1, HorizontalAlignment.Left);

lviewSession.Columns[0].Width = 100;

lviewSession.Columns[1].Width = 100;

lviewSession.Columns[2].Width = 50;

lviewSession.Columns[3].Width = 50;

lviewSession.Columns[4].Width = 200;

List of Access Files !!!!

lview.View = View.Details;

lview.Columns.Add("User Name", -1, HorizontalAlignment.Left);

lview.Columns.Add("Access Shared Folder",-1 , HorizontalAlignment.Left);

lview.Columns[0].Width = 200;

lview.Columns[1].Width = 200;

List of Created Files !!!!

lstCreate.View = View.Details;

lstCreate.Columns.Add("Created File Path", -1, HorizontalAlignment.Left);

lstCreate.Columns.Add("Date", -1, HorizontalAlignment.Left);

lstCreate.Columns.Add("Time", -1, HorizontalAlignment.Left);

lstCreate.Columns.Add("User", -1, HorizontalAlignment.Left);

lstCreate.Columns[0].Width = 200;

lstCreate.Columns[1].Width = 50;

lstCreate.Columns[2].Width = 50;

lstCreate.Columns[3].Width = 50;

List of Deleted Files

lstDelete.View = View.Details;

lstDelete.Columns.Add("Deleted File Path", -1, HorizontalAlignment.Left);

lstDelete.Columns.Add("Date", -1, HorizontalAlignment.Left);

lstDelete.Columns.Add("Time", -1, HorizontalAlignment.Left);

lstDelete.Columns[0].Width = 200;

lstDelete.Columns[1].Width = 50;

lstDelete.Columns[2].Width = 50;

List of Renamed Files

lstRenamed.View = View.Details;

lstRenamed.Columns.Add("Old File Path", -1, HorizontalAlignment.Left);

lstRenamed.Columns.Add("New File Path", -1, HorizontalAlignment.Left);

lstRenamed.Columns.Add("Date", -1, HorizontalAlignment.Left);

lstRenamed.Columns.Add("Time", -1, HorizontalAlignment.Left);

lstRenamed.Columns[0].Width = 150;

lstRenamed.Columns[1].Width = 150;

lstRenamed.Columns[2].Width = 50;

lstRenamed.Columns[3].Width = 50;

List of Changed Files

lstChanged .View = View.Details;

lstChanged.Columns.Add("File Path", -1, HorizontalAlignment.Left);

lstChanged.Columns.Add("Date", -1, HorizontalAlignment.Left);

lstChanged.Columns.Add("Time", -1, HorizontalAlignment.Left);

lstChanged.Columns[0].Width = 400;

lstChanged.Columns[1].Width = 200;

lstChanged.Columns[2].Width = 50;

dt.Columns.Add("Created File Path", Type.GetType("System.String"));

dt.Columns.Add("Date", Type.GetType("System.String"));

dt.Columns.Add("Time", Type.GetType("System.String"));

dt.Columns.Add("User", Type.GetType("System.String"));

dt1.Columns.Add("Old File Path", Type.GetType("System.String"));

dt1.Columns.Add("New File Path", Type.GetType("System.String"));

dt1.Columns.Add("Date", Type.GetType("System.String"));

dt1.Columns.Add("Time", Type.GetType("System.String"));

dt1.Columns.Add("User", Type.GetType("System.String"));

dt2.Columns.Add("Deleted File Path", Type.GetType("System.String"));

dt2.Columns.Add("Date", Type.GetType("System.String"));

dt2.Columns.Add("Time", Type.GetType("System.String"));

dt2.Columns.Add("User", Type.GetType("System.String"));

dt3.Columns.Add("File Path", Type.GetType("System.String"));

dt3.Columns.Add("Date", Type.GetType("System.String"));

dt3.Columns.Add("Time", Type.GetType("System.String"));

dt3.Columns.Add("User", Type.GetType("System.String"));

}

private void tabShared\_Click(object sender, EventArgs e)

{

}

private void load\_Shared\_Item(int cnt,string folder,string path, string Desc,string status)

{

listView1.Items.Add(folder);

listView1.Items[cnt].SubItems.Add(path);

listView1.Items[cnt].SubItems.Add(Desc);

listView1.Items[cnt].SubItems.Add(status);

}

private void load\_Access\_folder(int cnt, string username, string folder)

{

lview.Items.Add(username);

lview.Items[cnt].SubItems.Add(folder);

}

private void Read\_Shared\_Folder(string ip,string uname,string pass,bool cb)

{

int cnt = 0;

try

{

string ssWmiMachine1 = "\\\\" + "192.168.1.10" + "\\root\\cimv2";

MQL for Shaer Information

if(cb)

ip = System.Environment.MachineName;

if (!cb)

{

co = new ConnectionOptions();

co.EnablePrivileges = true;

co.Username = uname;

co.Password = pass;

co.Impersonation = System.Management.ImpersonationLevel.Impersonate;

}

string ssWmiMachine = "\\\\" + ip + "\\root\\cimv2";

ms = new ManagementScope(ssWmiMachine, co);

oq2 = new ObjectQuery("SELECT \* FROM Win32\_Share");

ManagementObjectSearcher searcher = new ManagementObjectSearcher(ms,oq2);

foreach (ManagementObject queryObj in searcher.Get())

{

RDirPath = queryObj["Path"].ToString();

if (!RDirPath.Equals("") || RDirPath.Equals("IPC$") || RDirPath.Equals("ADMIN$"))

{

File\_Watcher(RDirPath);

}

load\_Shared\_Item(cnt, queryObj["Name"].ToString(), queryObj["Path"].ToString(), queryObj["Description"].ToString(), queryObj["Status"].ToString());

cnt = cnt + 1;

}

}

catch (ManagementException e)

{

MessageBox.Show("An error occurred while querying for WMI data: " + e.Message);

}

}

Wll Load Current session Data

private void Load\_Current\_Session(int cnt, string ipaddress,string username, string accesstime,string ideltime,string os)

{

lviewSession.Items.Add(ipaddress);

lviewSession.Items[cnt].SubItems.Add(username);

lviewSession.Items[cnt].SubItems.Add(accesstime);

lviewSession.Items[cnt].SubItems.Add(ideltime);

lviewSession.Items[cnt].SubItems.Add(os);

}

private void Read\_Current\_Session()

{

int counter = 0;

int ActiveTime;

int itime;

string sharename;

try

{

oq = new System.Management.ObjectQuery("SELECT \* FROM Win32\_ServerConnection");

oq1 = new System.Management.ObjectQuery("SELECT \* FROM Win32\_ServerSession");

ManagementObjectSearcher searcher = new ManagementObjectSearcher(ms ,oq);

ManagementObjectSearcher searcher2 = new ManagementObjectSearcher(ms ,oq1);

Read the object

foreach (ManagementObject ServerQobj in searcher2.Get())

{

RemoteOS = ServerQobj["ClientType"].ToString();

IDealTime = ServerQobj["IdleTime"].ToString();

itime = Int32.Parse(IDealTime) 60;

IDealTime = itime.ToString();

}

usname = "";

foreach (ManagementObject queryObj in searcher.Get())

{

RemoteIPAddress = queryObj["ComputerName"].ToString();

RemoteUserName = queryObj["UserName"].ToString();

RemoteActiveTime = queryObj["ActiveTime"].ToString();

ActiveTime = (Int32.Parse(RemoteActiveTime)) 60;

RemoteActiveTime = ActiveTime.ToString();

sharename = queryObj["ShareName"].ToString();

if (!sharename.Equals("IPC$"))

{

if (usname != "")

{

usname = usname + " , " + RemoteUserName;

}

else

{

usname = RemoteUserName;

}

Load\_Current\_Session(counter, RemoteIPAddress, RemoteUserName, RemoteActiveTime,IDealTime,RemoteOS);

load\_Access\_folder(counter, RemoteUserName, sharename);

counter += 1;

}

}

}

catch (ManagementException e)

{

MessageBox.Show("An error occurred while querying for WMI data: " + e.Message);

this.Close();

}

}

private void Monitor\_Accessed\_Files()

{

}

private void listView1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void tbarRefresh\_Click(object sender, EventArgs e)

{

Refresh\_All();

}

private void statusStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void lviewSession\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

private void File\_Watcher(string sFolderPath)

{

if (Directory.Exists(sFolderPath))

{

FileSystemWatcher mywatcher = new FileSystemWatcher(sFolderPath);

mywatcher.Filter = "";

mywatcher.NotifyFilter = NotifyFilters.CreationTime | NotifyFilters.DirectoryName | NotifyFilters.FileName | NotifyFilters.LastAccess;

mywatcher.EnableRaisingEvents = true;

mywatcher.IncludeSubdirectories = true;

mywatcher.Created += new FileSystemEventHandler(mywatcher\_created);

mywatcher.Deleted += new FileSystemEventHandler(mywatcher\_deleted);

mywatcher.Changed += new FileSystemEventHandler(mywatcher\_changed);

mywatcher.Renamed += new RenamedEventHandler(mywatcher\_renamed);

if (iWatcherCount >= 100)

{

Array.Resize(ref mywatcher\_List, iWatcherCount + 1);

}

mywatcher\_List[iWatcherCount] = mywatcher;

iWatcherCount++;

}

}

protected void mywatcher\_created(object sender,FileSystemEventArgs e)

{

cCount = 0;

CheckForIllegalCrossThreadCalls = false;

DateTime current = DateTime.Now;

FileInfo fi = new FileInfo(e.FullPath.ToString());

if (fi.Extension != ".cmd" && fi.Extension != ".inf")

{

string ss = e.FullPath.ToString();

object[] va3 ={ ss, current.ToShortDateString(), current.ToShortTimeString(), usname };

try

{

DataRow dr;

dr = dt.NewRow();

dr[0] = ss;

dr[1] = current.ToShortDateString();

dr[2] = current.ToShortTimeString();

dr[3] = usname;

dt.Rows.Add(dr);

dataGridView1.DataSource = dt;

dataGridView1.Update();

}

catch { }

try

{

if (flage == true)

{

fs = new FileStream(@"c:\fcreated.txt", FileMode.Append, FileAccess.Write);

m\_streamWriter = new StreamWriter(fs);

m\_streamWriter.BaseStream.Seek(0, SeekOrigin.End);

flage = false;

}

m\_streamWriter.WriteLine(current.ToShortDateString() + " " + current.ToShortTimeString() + " " + "Created" + " "+ ss + " " + usname + "\n");

m\_streamWriter.Flush();

}

catch

{

m\_streamWriter.Close();

fs.Close();

flage = true;

}

cCount += 1;

}

}

protected void mywatcher\_renamed(object sender,RenamedEventArgs r)

{

cCount = 0;

CheckForIllegalCrossThreadCalls = false;

DateTime current = DateTime.Now;

FileInfo fi = new FileInfo(r.FullPath.ToString());

if (fi.Extension != ".cmd" && fi.Extension != ".inf")

{

try

{

DataRow dr3;

dr3 = dt1.NewRow();

dr3[0] = r.OldFullPath.ToString();

dr3[1] = r.FullPath.ToString();

dr3[2] = Convert.ToString( current.ToShortDateString());

dr3[3] = Convert.ToString(current.ToShortTimeString());

dr3[4] = usname;

dt1.Rows.Add(dr3);

dataGridView3.DataSource = dt1;

dataGridView3.Update();

}

catch { }

try

{

if (flager == true)

{

fs4 = new FileStream(@"c:\frenamed.txt", FileMode.Append, FileAccess.Write);

m\_streamWriter4 = new StreamWriter(fs);

m\_streamWriter4.BaseStream.Seek(0, SeekOrigin.End);

flager = false;

}

m\_streamWriter4.WriteLine(current.ToShortDateString() + " " + current.ToShortTimeString() + " " + "Renamed" + " " + r.OldFullPath.ToString() + " " + r.FullPath.ToString() + " " + usname + "\n");

m\_streamWriter4.Flush();

}

catch

{

m\_streamWriter4.Close();

fs4.Close();

flager = true;

}

rCount += 1;

}

}

protected void mywatcher\_deleted(object sender, FileSystemEventArgs e)

{

dCount = 0;

CheckForIllegalCrossThreadCalls = false;

DateTime current = DateTime.Now;

FileInfo fi = new FileInfo(e.FullPath.ToString());

if (fi.Extension != ".cmd" && fi.Extension != ".inf")

{

try

{

DataRow dr1;

dr1 = dt2.NewRow();

dr1[0] = e.FullPath.ToString();

dr1[1] = Convert.ToString(current.ToShortDateString());

dr1[2] = Convert.ToString(current.ToShortTimeString());

dr1[3] = usname;

dt2.Rows.Add(dr1);

dataGridView2.DataSource = dt2;

dataGridView2.Update();

}

catch { }

try

{

if (flaget == true)

{

fs2 = new FileStream(@"c:\fdeleted.txt", FileMode.Append, FileAccess.Write);

m\_streamWriter2 = new StreamWriter(fs);

m\_streamWriter2.BaseStream.Seek(0, SeekOrigin.End);

flaget = false;

}

m\_streamWriter2.WriteLine(current.ToShortDateString() + " " + current.ToShortTimeString() + " " + "Deleted" + " "+e.FullPath.ToString() + " " + usname + "\n");

m\_streamWriter2.Flush();

}

catch

{

m\_streamWriter2.Close();

fs2.Close();

flaget = true;

}

}

}

public void checkdata(object o,DataGridViewDataErrorEventArgs de)

{

MessageBox.Show("Data gridview Data Error"+de.RowIndex +" "+de.Context );

de.Cancel = true;

}

protected void mywatcher\_changed(object sender, FileSystemEventArgs e)

private void TaskBar\_Initilization()

{

taskbarNotifier1.SetBackgroundBitmap(new Bitmap(GetType(), "skin"), Color.FromArgb(255, 0, 255));

taskbarNotifier1.SetBackgroundBitmap("Skin.bmp", Color.FromArgb(255, 0, 255));

taskbarNotifier1.SetCloseBitmap(new Bitmap(GetType(), "close"), Color.FromArgb(255, 0, 255), new Point(127, 8));

taskbarNotifier1.SetCloseBitmap("close.bmp", Color.FromArgb(255, 0, 255), new Point(127, 8));

taskbarNotifier1.TitleRectangle = new Rectangle(40, 9, 70, 25);

taskbarNotifier1.ContentRectangle = new Rectangle(8, 41, 133, 68);

taskbarNotifier1.TitleClick += new EventHandler(TitleClick);

taskbarNotifier1.ContentClick += new EventHandler(ContentClick);

taskbarNotifier1.CloseClick += new EventHandler(CloseClick);

}

void CloseClick(object obj, EventArgs ea)

{

MessageBox.Show("Closed was Clicked");

}

void TitleClick(object obj, EventArgs ea)

{

MessageBox.Show("Title was Clicked");

}

void ContentClick(object obj, EventArgs ea)

{

MessageBox.Show("Content was Clicked");

}

private void button1\_Click(object sender, EventArgs e)

{

}

void Show\_popUP()

{

string t1 = "500";

string t2 = "3000";

string t3 = "500";

taskbarNotifier1.CloseClickable = true;

taskbarNotifier1.TitleClickable = false;

taskbarNotifier1.ContentClickable = true;

taskbarNotifier1.EnableSelectionRectangle = true;

taskbarNotifier1.KeepVisibleOnMousOver = true; Added Rev 002

taskbarNotifier1.ReShowOnMouseOver = true; Added Rev 002

taskbarNotifier1.Show("Monitoring", RemoteUserName + "\n Is Now Accessing Your System ", Int32.Parse(t1), Int32.Parse(t2), Int32.Parse(t3));

}

private void Auto\_Refresh()

{

tim.Start();

tim.Tick+=new EventHandler(tim\_Tick);

}

void tim\_Tick(object obj, EventArgs ea)

{

Timcnt+=1;

newcount = lviewSession.Items.Count;

if (oldcount != newcount)

{

oldcount = newcount;

}

if (Timcnt > 100\*\_RefreshTime)

{

Refresh\_All();

oldcount = newcount;

Timcnt = 0;

}

}

private void Refresh\_All()

{

int len;

len = RemoteUserName.Length;

if(len>0 )

{

if(\_ShowPopUp)

Show\_popUP();

}

for (int i = 0; i < iWatcherCount; i++)

mywatcher\_List[i].Dispose();

listView1.Items.Clear();

lviewSession.Items.Clear();

lview.Items.Clear();

if (Ostate)

{

try

{

Read\_Shared\_Folder(ip.Text, uname.Text, pass.Text, cb.Checked);

Read\_Current\_Session();

}

catch { }

}

}

private void settingsToolStripMenuItem1\_Click\_1(object sender, EventArgs e)

{

if (this.settingsToolStripMenuItem1.Checked == true)

{

\_ShowPopUp = false;

this.settingsToolStripMenuItem1.Checked = false;

}

else

{

\_ShowPopUp = true;

this.settingsToolStripMenuItem1.Checked = true;

}

MessageBox.Show(\_ShowPopUp.ToString());

}

private void settingsToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void toolStripMenuItem6\_Click(object sender, EventArgs e)

{

toolStripMenuItem6.Checked = true;

toolStripMenuItem7.Checked = false ;

toolStripMenuItem8.Checked = false;

toolStripMenuItem9.Checked = false;

toolStripMenuItem10.Checked = false;

\_RefreshTime = 5;

}

private void toolStripMenuItem7\_Click(object sender, EventArgs e)

{

toolStripMenuItem6.Checked = false;

toolStripMenuItem7.Checked = true;

toolStripMenuItem8.Checked = false;

toolStripMenuItem9.Checked = false;

toolStripMenuItem10.Checked = false;

\_RefreshTime = 10;

}

private void toolStripMenuItem8\_Click(object sender, EventArgs e)

{

toolStripMenuItem6.Checked = false;

toolStripMenuItem7.Checked = false;

toolStripMenuItem8.Checked = true;

toolStripMenuItem9.Checked = false;

toolStripMenuItem10.Checked = false;

\_RefreshTime = 25;

}

private void toolStripMenuItem9\_Click(object sender, EventArgs e)

{

toolStripMenuItem6.Checked = false ;

toolStripMenuItem7.Checked = false;

toolStripMenuItem8.Checked = false;

toolStripMenuItem9.Checked = true;

toolStripMenuItem10.Checked = false;

\_RefreshTime = 30;

}

private void toolStripMenuItem10\_Click(object sender, EventArgs e)

{

toolStripMenuItem6.Checked = false;

toolStripMenuItem7.Checked = false;

toolStripMenuItem8.Checked = false;

toolStripMenuItem9.Checked = false;

toolStripMenuItem10.Checked = true;

\_RefreshTime = 50;

}

private void aboutMeToolStripMenuItem\_Click(object sender, EventArgs e)

{

About abt = new About();

abt.ShowDialog();

}

private void helpToolStripMenuItem1\_Click(object sender, EventArgs e)

{

MessageBox.Show("VerySimple To Use", "Monitoring", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

<summary>

Click on Clear Log History

<summary>

<param name="sender"><param>

<param name="e"><param>

private void Right\_ClearLogHistry\_Click(object sender, EventArgs e)

{

switch (\_CurrentActiveTab)

{

case 1:

ClearLogHistory(1);

break;

case 2:

ClearLogHistory(2);

break;

case 3:

ClearLogHistory(3);

break;

case 4:

ClearLogHistory(4);

break;

}

}

<summary>

Clearing The History of Current Grid

<summary>

<param name="\_ActiveTab">Current Selected Grid<param>

protected void ClearLogHistory(int \_ActiveTab)

{

int Count;

switch (\_ActiveTab)

{

case 1: Clear The Create Grid List

Count = lstCreate.Items.Count;

while (Count != 0)

{

Count--;

lstCreate.Items[Count].Remove();

}

cCount = 0;

break;

case 2: Clear The Delete Grid List

Count = lstDelete.Items.Count; for (int i = 0; i < Count; i++)

while (Count != 0)

{

Count--;

lstDelete.Items[Count].Remove();

}

dCount = 0;

break;

case 3: Clear The Renamed Grid List

Count = lstRenamed.Items.Count;

while (Count != 0)

{

Count--;

lstRenamed.Items[Count].Remove();

}

rCount = 0;

break;

case 4: Clear The Changed Grid List

Count = lstChanged.Items.Count;

while (Count != 0)

{

Count--;

lstChanged.Items[Count].Remove();

}

cHCount = 0;

break;

}

}

<summary>

Check The Right Click on Create File List Grid

<summary>

<param name="sender"><param>

<param name="e"><param>

private void lstCreate\_MouseDown(object sender, MouseEventArgs e)

{

if(e.Button==MouseButtons.Right)

\_CurrentActiveTab = 1;

}

<summary>

Check Right Click on File Delete Grid

<summary>

<param name="sender"><param>

<param name="e"><param>

private void lstDelete\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

\_CurrentActiveTab = 2;

}

private void lstChanged\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

\_CurrentActiveTab = 4;

}

<summary>

Right Click for check the current Active Accessing USer

<summary>

<param name="sender"><param>

<param name="e"><param>

private void RightCurrentUser\_Click(object sender, EventArgs e)

{

int Count;

string \_ActiveUser="Current Active IP :\n";

Count = lviewSession.Items.Count;

while (Count != 0)

{

Count--;

\_ActiveUser+= lviewSession.Items[Count].Text.ToString() + "\n";

}

MessageBox.Show(\_ActiveUser, "Monitoring your Pc", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void RightExportToFile\_Click(object sender, EventArgs e)

{

switch (\_CurrentActiveTab)

{

case 1:

ExportToTextFile(1);

break;

case 2:

ExportToTextFile(2);

break;

case 3:

ExportToTextFile(3);

break;

case 4:

ExportToTextFile(4);

break;

}

}

<summary>

public void ExportToTextFile(int \_ActiveTab)

{

int Count;

string \_Result="";

bool \_ClearFlag = false;

switch(\_ActiveTab)

{

case 1:

Count=lstCreate.Items.Count;

if (Count != 0)

{

TextWriter Tw = new StreamWriter(\_CreateLogFilePath);

Tw.WriteLine("Created FilePath" + "\t" + "Date" + "\t" + "Time");

Tw.WriteLine("-----------------------------------------------------------------");

for (int j = 0; j < Count; j++)

{

ListViewItem lvi = lstCreate.Items[j];

for (int i = 0; i < lvi.SubItems.Count; i++)

{

\_Result += lvi.SubItems[i].Text + "\t";

}

Tw.WriteLine(\_Result);

\_Result = "";

}

Tw.Close();

\_Result = "";

\_ClearFlag = true;

}

else

{

MessageBox.Show("Nothing to store in log file !", "NWM: Logging", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

\_ClearFlag = false;

}

break;

case 2:

Count = lstDelete.Items.Count;

if (Count != 0)

{

TextWriter Tw1 = new StreamWriter(\_DeleteLogFilePath);

Tw1.WriteLine("Deleted FilePath" + "\t" + "Date" + "\t" + "Time");

Tw1.WriteLine("-----------------------------------------------------------------");

for (int j = 0; j < Count; j++)

{

ListViewItem lvi = lstDelete.Items[j];

for (int i = 0; i < lvi.SubItems.Count; i++)

{

\_Result += lvi.SubItems[i].Text + "\t";

}

Tw1.WriteLine(\_Result);

\_Result = "";

}

Tw1.Close();

\_Result = "";

\_ClearFlag = true;

}

else

{

MessageBox.Show("Nothing to store in log file !", "NWM: Logging", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

\_ClearFlag = false;

}

break;

case 3:

Count = lstRenamed.Items.Count;

if (Count != 0)

{

TextWriter Tw2 = new StreamWriter(\_RenamedLogFilePath);

Tw2.WriteLine("Old FilePath" + "\t" + "New File Path" + "\t" + "Date" + "\t" + "Time");

Tw2.WriteLine("-----------------------------------------------------------------");

for (int j = 0; j < Count; j++)

{

ListViewItem lvi = lstRenamed.Items[j];

for (int i = 0; i < lvi.SubItems.Count; i++)

{

\_Result += lvi.SubItems[i].Text + "\t";

}

Tw2.WriteLine(\_Result);

\_Result = "";

}

Tw2.Close();

\_Result = "";

\_ClearFlag = true;

}

else

{

MessageBox.Show("Nothing to store in log file !", "NWM: Logging", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

\_ClearFlag = false;

}

break;

case 4:

Count = lstChanged.Items.Count;

if (Count != 0)

{

TextWriter Tw3 = new StreamWriter(\_ChangedLogFilePath);

Tw3.WriteLine("Changed FilePath" + "\t" + "Date" + "\t" + "Time");

Tw3.WriteLine("-----------------------------------------------------------------");

for (int j = 0; j < Count; j++)

{

ListViewItem lvi = lstChanged.Items[j];

for (int i = 0; i < lvi.SubItems.Count; i++)

{

\_Result += lvi.SubItems[i].Text + "\t";

}

Tw3.WriteLine(\_Result);

\_Result = "";

}

Tw3.Close();

\_Result = "";

\_ClearFlag = true;

}

else

{

MessageBox.Show("Nothing to store in log file !", "NWM: Logging", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

\_ClearFlag = false;

}

break;

}

if (\_ClearFlag == true)

{

DialogResult Result = MessageBox.Show("Log has been written successfuly !\n Do you want to clear this Grid ?", "NWM : Loging", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (Result == DialogResult.Yes)

{

Clear the Grid

ClearLogHistory(\_CurrentActiveTab);

}

}

}

private void listBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.Close();

}

private void clearLogHistoryToolStripMenuItem\_Click(object sender, EventArgs e)

{

ClearLog();

}

public void ClearLog()

{

dt.Clear();

dt1.Clear();

dt2.Clear();

dt3.Clear();

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

Ostate = true;

button1.Visible = true;

button1.Enabled = true;

button1.Text = "Logout";

toolStripButton1.Enabled = false;

call();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

Ostate = false;

toolStripButton1.Enabled = true;

ListView\_init();

listView1.Items.Clear();

lviewSession.Items.Clear();

lview.Items.Clear();

}

}

}

public int getMaxNetworkHosts ()

{

// 2^(32-networkprefix)

// -2 ... because .0 and .255

return ((int)Math.Pow(2,(32-this.networkPrefix)))-2;

}

public int getMaxSubNets ()

{

// Bits from Subnet = prefix-class\_prefix

int count = (int)Math.Pow(2, this.networkPrefix - this.getNetworkClassPrefix());

// -2 because 1 bit for routing

if (!this.isFirstSubNetBit() || this.getNetworkClassPrefix()==this.networkPrefix)

count-=2;

if (count < 0 ) count = 0;

return count;

}

public long getNetworkLong ()

{

long mask = (long)Math.Pow (2, this.networkPrefix)-1;

mask = mask << (32-this.networkPrefix);

return (mask & ip2Long());

}

public String getNetwork ()

{

return long2String(getNetworkLong());

}

public long getBroadCastLong ()

{

long netMask = (long)Math.Pow (2, this.networkPrefix)-1;

netMask = netMask << (32-this.networkPrefix);

long hostMask = (long) Math.Pow(2, 32-this.networkPrefix)-1;

return (netMask = (ip2Long() & netMask) | hostMask);

}

public String getBroadCast ()

{

return long2String(getBroadCastLong());

}

public String[] getNetworkIPRange ()

{

String [] result = new String [2];

//String from, to;

/\*\*

\* Start

\* +1 because first = network

\*/

result [0] = long2String (getNetworkLong() + 1);

/\*\*

\* End

\* -1 because last = broadcast

\*/

result [1] = long2String(getBroadCastLong() -1);

return result;

}

public long ip2Long ()

{

return ((long)(IPTool.byte2int(this.get1Byte()) \* 256 +

IPTool.byte2int(this.get2Byte())) \* 256 +

IPTool.byte2int(this.get3Byte())) \* 256 +

IPTool.byte2int(this.get4Byte());

}

public String long2String (long ip)

{

long a = (long)(ip & 0xff000000) >> 24;

long b = (long)(ip & 0x00ff0000) >> 16;

long c = (long)(ip & 0x0000ff00) >> 8;

long d = (long)(ip & 0xff);

return a+"."+b+"."+c+"."+d;

}

public bool isFirstSubNetBit()

{

return firstSubNetBit;

}

public void setFirstSubNetBit(bool b)

{

firstSubNetBit = b;

}

public static IPTool getNextIP (String ip)

{

IPTool nextIp = new IPTool (ip);

nextIp.setIp(nextIp.long2String(nextIp.ip2Long()+(long)1));

return nextIp;

}

public IPTool getNextSubNet (long numberOfHosts)

{

IPTool ip = IPTool.getNextIP(this.getBroadCast());

ip.setIp(this.long2String(ip.ip2Long()+(long)1));

ip.setFirstSubNetBit(this.isFirstSubNetBit());

int lastPrefix = ip.getNetworkClassPrefix();

int prefix = 30;

do

{

if (prefix < ip.getNetworkClassPrefix())

return null; // no subnet found

ip.setNetworkPrefix(prefix);

prefix--;

// ignore subnetbit?

if (!ip.isFirstSubNetBit() && prefix == lastPrefix+1) prefix--;

}

while (ip.getMaxNetworkHosts() < numberOfHosts);

return ip;

}

}

}

// Attache Email

namespace Networks

{

public partial class frmAddAttachment : DevComponents.DotNetBar.Office2007Form

{

public frmAddAttachment()

{

InitializeComponent();

}

private void btnBrowse\_Click(object sender, EventArgs e)

{

dlg.ShowDialog();

txtFile.Text = dlg.FileName;

}

private void btnCancel\_Click(object sender, EventArgs e)

{

txtFile.Text = "";

this.Hide();

}

private void btnOK\_Click(object sender, EventArgs e)

{

this.Hide();

}

private void frmAddAttachment\_Load(object sender, EventArgs e)

{

this.Office2007ColorTable = DevComponents.DotNetBar.Rendering.eOffice2007ColorScheme.Black ;

}

}

}

namespace Networks

{

public partial class Email : DevComponents.DotNetBar.Office2007Form

{

MailMessage mail = new MailMessage();

public Email()

{

InitializeComponent();

}

private void BtnAdd\_Click(object sender, EventArgs e)

{

Networks.frmAddAttachment frm = new Networks.frmAddAttachment();

frm.ShowDialog(this);

if (frm.txtFile.Text.Trim() != "")

listBox1.Items.Add(frm.txtFile.Text);

frm.Dispose();

}

private void BtnRemove\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex > -1)

listBox1.Items.RemoveAt(listBox1.SelectedIndex);

}

private void BtnSend\_Click(object sender, EventArgs e)

{

try

{

using (MailMessage mailMessage = new MailMessage(new MailAddress(textBox1.Text),

new MailAddress(textBox1.Text)))

{

mailMessage.Body = textBox5.Text;

mailMessage.Subject = textBox4.Text;

try

{

SmtpClient SmtpServer = new SmtpClient();

SmtpServer.Credentials = new System.Net.NetworkCredential(textBox2.Text, textBox3.Text);

SmtpServer.Port = 587;

if (checkBox1.Checked == true)

try

{

SmtpServer.Host = "smtp.gmail.com";

SmtpServer.EnableSsl = true;

}

catch

{

SmtpServer.Host = "smtp.gmail.com";

SmtpServer.EnableSsl = true;

}

if (checkBox2.Checked == true)

try

{

SmtpServer.Host = "smtp.mail.yahoo.com";

SmtpServer.EnableSsl = false;

}

catch

{

SmtpServer.Host = "smtp.mail.yahoo.com";

SmtpServer.EnableSsl = false;

}

if (checkBox3.Checked == true)

try

{

SmtpServer.Host = "smtp.aol.com";

SmtpServer.EnableSsl = false;

}

catch

{

SmtpServer.Host = "smtp.aol.com";

SmtpServer.EnableSsl = false;

}

if (checkBox4.Checked == true)

try

{

SmtpServer.Credentials = new System.Net.NetworkCredential(textBox2.Text, textBox3.Text);

SmtpServer.Host = "smtp.live.com";

SmtpServer.EnableSsl = true;

}

catch

{

SmtpServer.Credentials = new System.Net.NetworkCredential(textBox2.Text, textBox3.Text);

SmtpServer.Host = "smtp.live.com";

SmtpServer.EnableSsl = true;

}

if (checkBox1.Checked && checkBox2.Checked == true)

try

{

MessageBox.Show("Please Choose Only One Email Client At A Time");

}

catch

{

MessageBox.Show("Please Choose Only One Email Client At A Time");

}

mail = new MailMessage();

String[] addr = textBox1.Text.Split(',');

mail.From = new MailAddress(textBox2.Text);

Byte i;

for (i = 0; i < addr.Length; i++)

mail.To.Add(addr[i]);

mail.Subject = textBox4.Text;

mail.Body = textBox5.Text;

if (listBox1.Items.Count != 0)

{

for (i = 0; i < listBox1.Items.Count; i++)

mail.Attachments.Add(new Attachment(listBox1.Items[i].ToString()));

}

mail.IsBodyHtml = true;

mail.DeliveryNotificationOptions = DeliveryNotificationOptions.OnFailure;

mail.ReplyTo = new MailAddress(textBox1.Text);

SmtpServer.Send(mail);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "EMail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "EMail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

private void Email\_Load(object sender, EventArgs e)

{

this.Office2007ColorTable = DevComponents.DotNetBar.Rendering.eOffice2007ColorScheme.Black;

}

private void groupPanel3\_Click(object sender, EventArgs e)

{

}

}

}

namespace Networks

{

public partial class RemoteDesktop : DevComponents.DotNetBar.Office2007Form

{

public RemoteDesktop()

{

InitializeComponent(); }

private void RemoteDesktop\_Load(object sender, EventArgs e)

{

this.Office2007ColorTable = DevComponents.DotNetBar.Rendering.eOffice2007ColorScheme.Black;

}

private void BtnConnect\_Click(object sender, EventArgs e)

{

try

{

rdp.Server = txtServer.Text;

rdp.UserName = txtUserName.Text;

IMsTscNonScriptable secured = (IMsTscNonScriptable)rdp.GetOcx();

secured.ClearTextPassword = txtPassword.Text;

rdp.Connect();

}

catch (Exception Ex)

{

MessageBox.Show("Error Connecting", "Error connecting to remote desktop " + txtServer.Text + " Error: " + Ex.Message, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void BtnDisconnect\_Click(object sender, EventArgs e)

{

try

{

// Check if connected before disconnecting

if (rdp.Connected.ToString() == "1")

rdp.Disconnect();

}

catch (Exception Ex)

{

MessageBox.Show("Error Disconnecting", "Error disconnecting from remote desktop " + txtServer.Text + " Error: " + Ex.Message, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**CHAPTER 8**

**SYSTEM TESTING**

System testing involves user training system testing and successful running of the developed proposed system. The user tests the developed system and changes are made according to their needs. The testing phase involves the testing of developed system using various kinds of data.

An elaborate testing of data is prepared and the system is tested using the test data. While testing, errors are noted and the corrections are made. The corrections are also noted for the future use. The users are trained to operate the developed system.

# TESTING

System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes logical assumption that if all the parts of the system are correct, then the goal will be successfully achieved. A series of testing are done for the proposed system before the system is ready for the user acceptance testing.

# The following are the types of Testing

# Unit Testing

# Integration Testing

1. Validation Testing
2. Verification testing
3. User acceptance testing

# 9.1 Unit Testing

Unit testing focuses verification efforts on the smallest unit of the software design, the module. This is also known as “module testing”. The modules are tested separately. This testing was carried out during programming stage itself. In this testing each module is found to be working satisfactorily as regards to the expected output from the module.

**9.2 Integration Testing**

Data can be lost across an interface: one module can have adverse efforts on another. Integration testing is the systematic testing for construction of program structure, while at the same time conducting tests to uncover errors associated within the interface. Here correction is difficult because the isolation of cause is complicated by the cast expense of the entire program. Thus in the integration testing step, all the errors uncovered are corrected for the next testing steps.

# 9.3 Validation Testing

At the conclusion of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins validation test has been conducted one of the two possible conditions exists. One is the function or performance characteristics confirm to specification and are accepted and the other is deviation from specification is uncovered and a deficiency list is created.

**9.4 Verification Testing**

Verification is a fundamental concept in software design. This is the bridge between customer requirements and an implementation that satisfies those requirements.

This is verifiable if it can be demonstrated that the testing will result in an implementation that satisfies the customer requirements.

# 9.5 User Acceptance Testing

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

**CHAPTER 9**

**PROBLEMS FACED**

When there is a clear goal in sight but no clear set of directions or means to attain that goal, then it is called a problem. problems can be broken down into four aspects; goal, givens, means of transforming conditions, and obstacles.

**Goal –** the goal is the desired end state which the problem solving is being directed toward.

The hope is to reach that end state and be able to assess whether or not you achieved what you wanted.

**Givens-** these are the objects , conditions ,and constraints that accompany a problem ,and can be either explicit or implicit.

**Means of transforming conditions**- there should be a way of changing the initial stateof the problem.this is most usually a person’s knowledge or skill level. For instance ,a computer programmer presented with a problem would utilize his or her knowledge of programming language to transform the state of the problem.

**Obstacles**- the problem should present a challenge.if there are no challenges involved and the situation can be easily solved then it is not so a problem so much as a rountine task.

Every problem has a **problem faced**, which is the whole range of possible states and operators.only some of these states and operators will bring the person closer to the goal state. The problem starts at the **initial state** and **operators** are applied to change the state, creating a series of intermediate states that should hopefully lead to the final goal state

**CHAPTER 10**

**FUTURE PLANS**

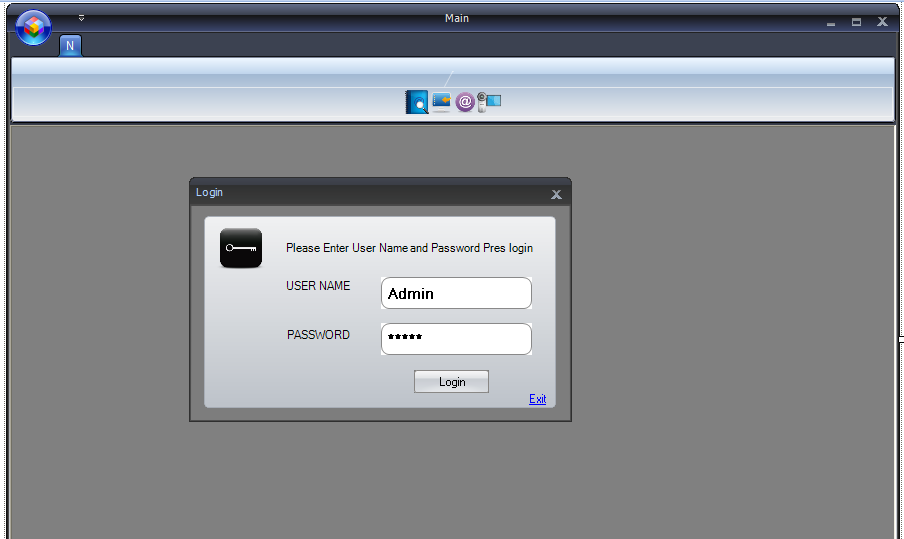
Every application has its own merits and demerits. The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Changing the existing modules or adding new modules can append improvements. Further enhancements can be made to the application, so that the web site functions very attractive and useful manner than the present one

CONCLUSION

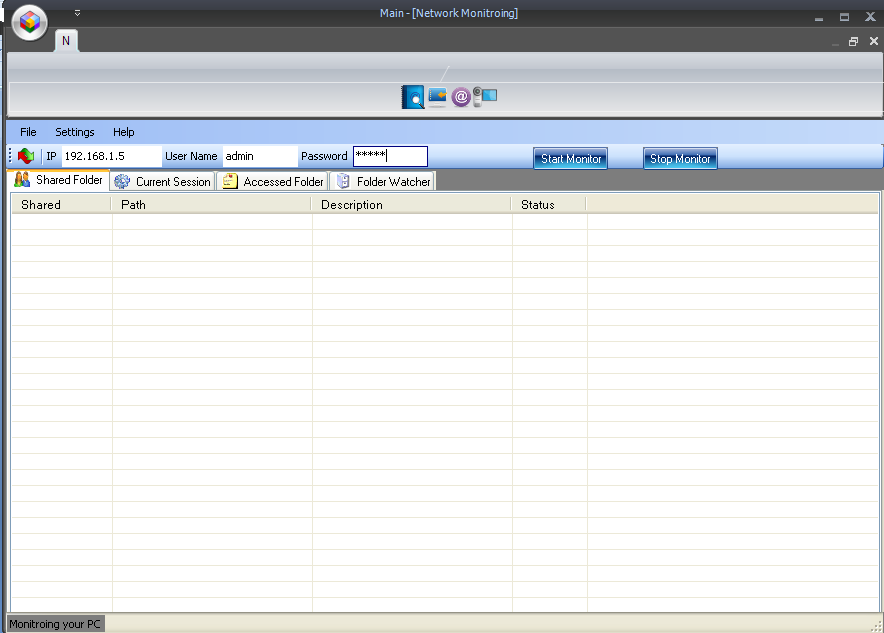
It is concluded that the application works well and satisfy the company and students. The application is tested very well and errors are properly debugged. The application is simultaneously accessed from more than one system. Simultaneous login from more than one place is tested.

The application works according to the restrictions provided in their respective browsers. Further enhancements can be made to the application, so that the application functions very interactive and useful to existing application .The application satisfies both the company and students by eliminating more input. The speed of the transactions become more enough now.

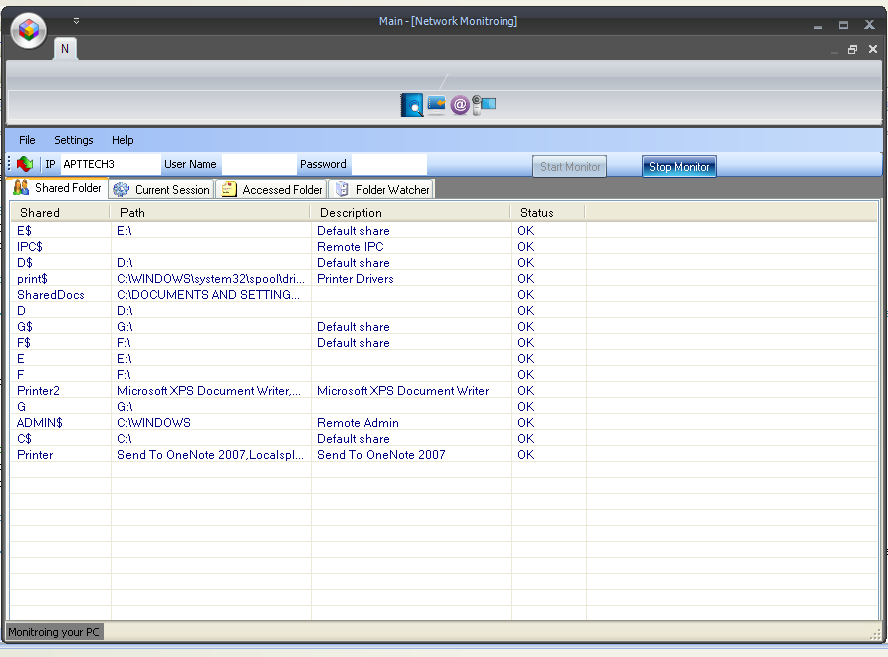
**APPENDIX**

****

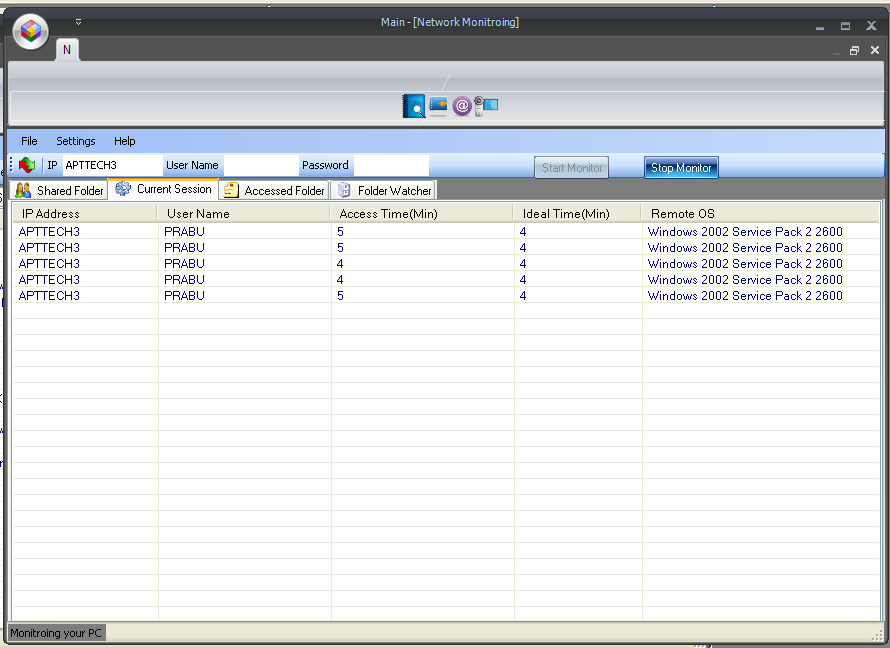
**Fig1.Login**

****

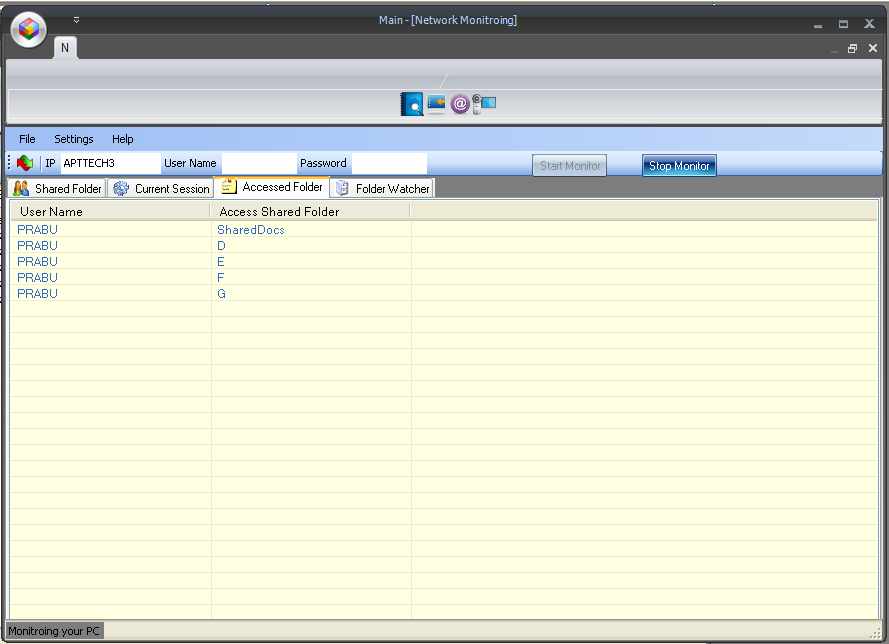
**Fig2.Main\_network**

****

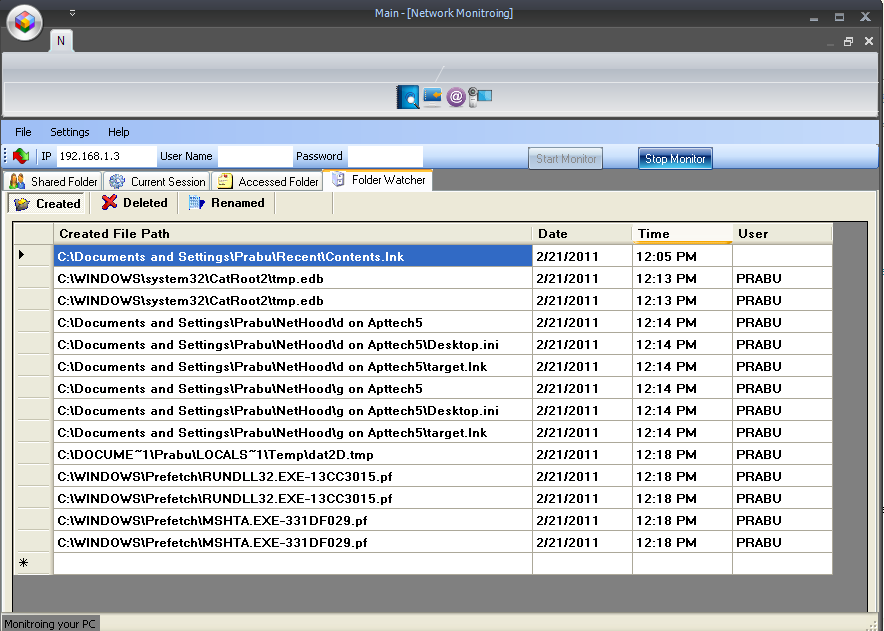
**Fig3.Sharefolder**

****

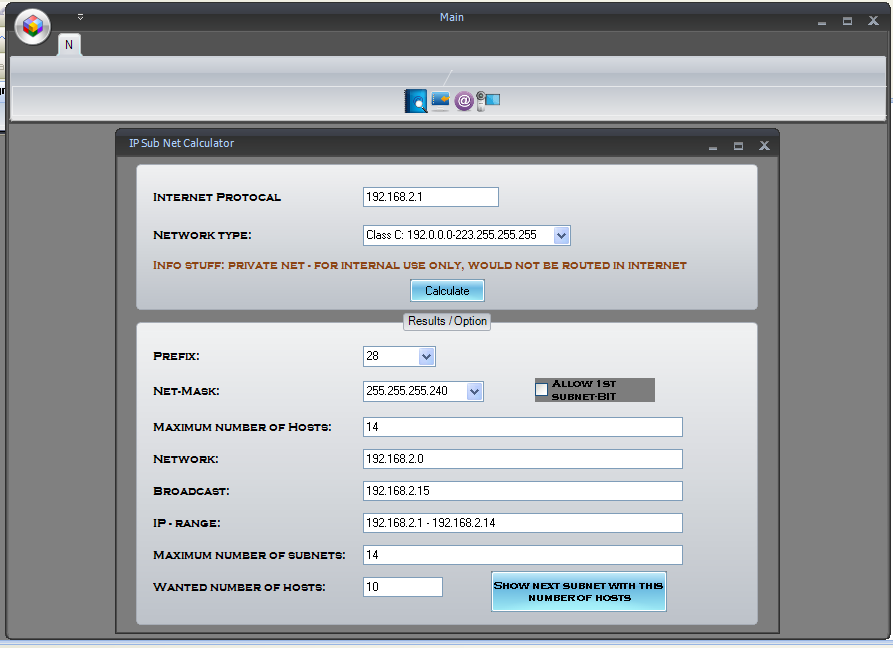
**Fig3. Network-Monitoring**

****

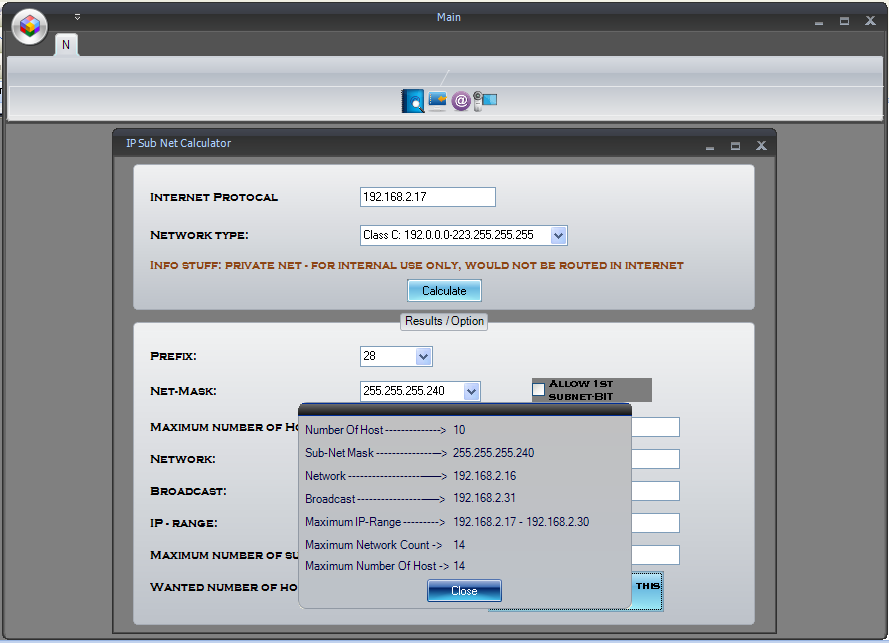
**Fig4.Current Session**

****

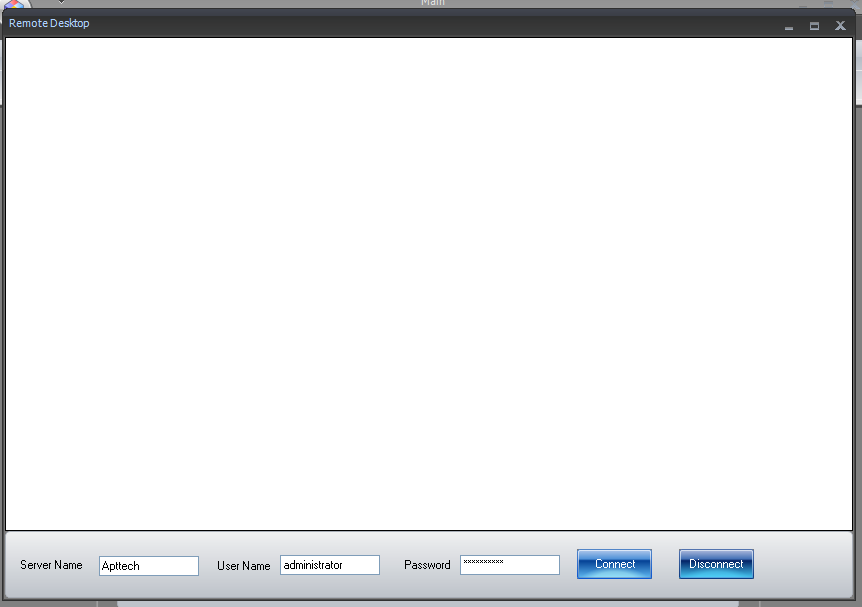
**Fig 5.Folder Watcher**

****

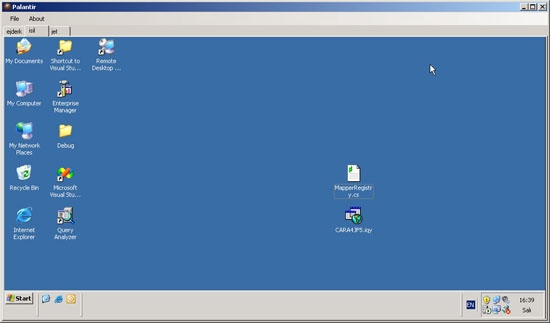
**Fig6.Ipsubnet**

****

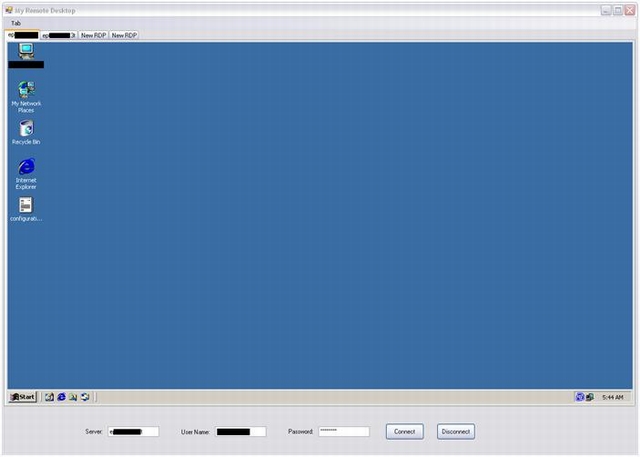
**Fig7.Ipsubnet mask calculation**

****

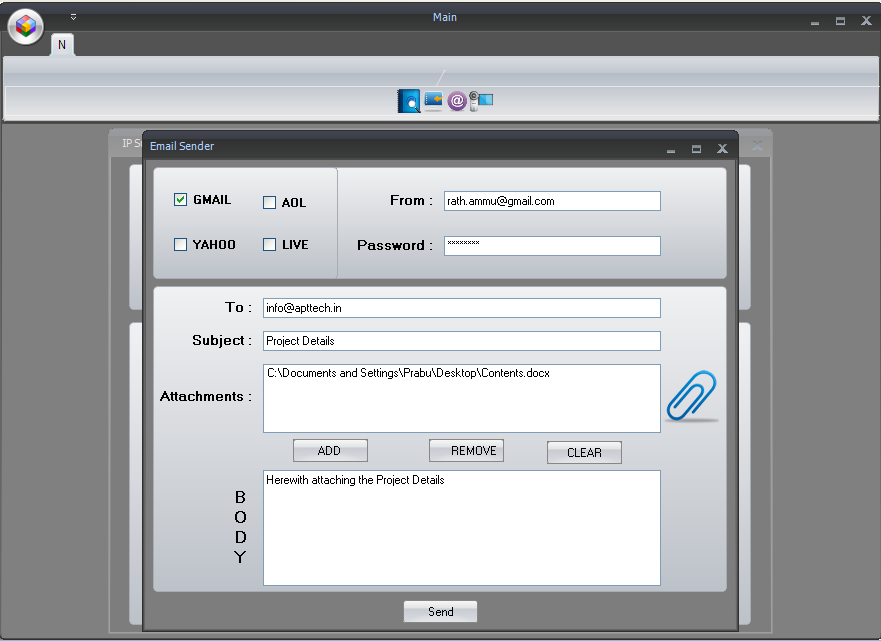
**Fig8.Remote Desktop**

****

**Fig9.Server 1**

****

**Fig10.Server 2**

****

**Fig11.E-Mail Client**

**REFERENCES**

Book Reference

* Professional ASP.NET MVC 1.0 (Wrox Programmer to Programmer) Rob Conery, Scott Hanselman, Phil Haack, Scott Guthrie Publisher: Wrox
* ASP.NET 3.5 Unleashed Stephen Walther Publisher: Sams
* Programming ASP.NET 3.5 Jesse Liberty, Dan Maharry, Dan Hurwitz Publisher: O'Reilly Media, Inc.

BIBLIOGRAPHY

* http:www.15seconds.comissue041020.htm
* http:www.a1vbcode.comapp-3619.asp
* http:www.aspcode.netASPNET-301-redirect.aspx
* http:www.aspcode.netMaster-pages-in-ASP-free-template-engine.aspx