**INTRODUCTION**

The project ‘**Server Monitoring’** is aimed at developing an application that is able to test and verify different services of a server and ensure that all of them are being delivered to end user adequately. It is developed in order to help the businesses to ensure that their network is live and responding. Monitoring a server means that the server owner always knows if one or all of his services go down.

Server monitoring may be **internal**, i.e. web server software checks its status and notifies the owner if some services go down, and **external**, i.e. some web server monitoring companies check the services status with a certain frequency. External monitoring is much more reliable, as it keeps on working when the server completely goes down.

**Server Monitoring** is composed of four modules:

1) Extensive Network Monitoring

2) Memory and Performance Management

3) System Snapshot

4) Flexible Alerting Engine

High **network monitoring** includes:

* **TCP Services** - Any services that will respond to a TCP connection, such as web, ftp, and mail servers.
* **DNS**- Get the correct IP address back from the DNS server.
* **Web Services** - Verify the status and content of your web servers (HTTP, HTTPS).

The module **Extensive Network Monitoring** is aimed at developing an application that is able to test and verify whether the end users can interact with the website or web application. It is developed in order to help the businesses to ensure that their website is

live and responding. It also helps in reducing the downtime of websites and makes them always available to end-users. Apart from monitoring the status of the website, it also finds the IP address of a given domain name and checks the availability of the port for it.

When an IP packet is to be forwarded, a router uses its forwarding table to determine the next hop for the packet's destination (based on the destination IP address in the IP packet header), and forwards the packet appropriately. The next router then repeats this process using its own forwarding table, and so on until the packet reaches its destination. At each stage, the IP address in the packet header is sufficient information to determine the next hop; no additional protocol headers are required.

**Memory and Performance Management** is the process of determining the network resources required to prevent a performance or availability impact on business-critical applications. Performance management is the practice of managing network service response time, consistency, and quality for individual and overall services. Performance management includes activities that ensure that goals are consistently being met in an effective and efficient manner.

**System Snapshot** is composed of two modules:

1) Fetching client system details on server

2) Fetching current system details.

All the clients’ details associated with a server can be fetched. Client details include their system details in terms of software’s installed, services installed etc. These details are generated in form of a report of specified format.

**Flexible Alerting Engine** will be designed for real life alerting with mail alerting types. **SMTP mail** with or without authentication and with fall-back to a 2ndare sent to the end user. It alerts the user about the system or about the web page on a regular time interval.

**REQUIREMENT ANALYSIS**

**2.1 INTRODUCTION**

The project **Server Monitoring** is aimed at developing an application that is able to test and verify different services of a server and ensure that all of them are being delivered to end user adequately. It is developed in order to help the businesses to ensure that their network is live and responding. Monitoring a server means that the server owner always knows if one or all of his services go down.

Server monitoring may be **internal**, i.e. web server software checks its status and notifies the owner if some services go down, and **external**, i.e. some web server monitoring companies check the services status with a certain frequency. External monitoring is much more reliable, as it keeps on working when the server completely goes down.

**Network monitoring** companies that offer website performance monitoring allow businesses to simulate the actions of thousands of visitors to a website and observe how it responds. They also simulate visitors across multiple geographies and servers Internet connections. Performance monitoring tools send out alerts when pages or parts of a website malfunction, which allows the [webmaster](http://en.wikipedia.org/wiki/Webmaster) to correct issues faster.

Network monitoring allows interested parties to track the health of a website or web application. A software program can periodically check to see if a website is down, if there are broken links or if errors have occurred on particular pages. For example, a web developer who hosts and maintains a website for a customer may want to be notified instantly when the site goes or when one of the web applications returns an error.

**IP Address Finder** helps user to find the IP address of a given website. The user enters the URL and the IP address of the URL is provided by the IP Address Finder. IP address of any entered valid Domain Name is given. Port Availability allows users to find which

ports are occupied on a system and which are free for the domain. Any domain name and its port is entered and its availability is checked on the system and result is displayed.

When an IP packet is to be forwarded, a router uses its forwarding table to determine the next hop for the packet's destination (based on the destination IP address in the IP packet header), and forwards the packet appropriately. The next router then repeats this process using its own forwarding table, and so on until the packet reaches its destination. At each stage, the IP address in the packet header is sufficient information to determine the next hop; no additional protocol headers are required.

**Memory and Performance Management** is the process of determining the network resources required to prevent a performance or availability impact on business-critical applications. Performance management is the practice of managing network service response time, consistency, and quality for individual and overall services. Performance management includes activities that ensure that goals are consistently being met in an effective and efficient manner.

**System snapshot** is developed to provide facility to their end users i.e. employees of making a software request if required.

It is composed of two modules:

1) Fetching client system details on server

2) Fetching current system details.

All the clients’ details associated with a server can be fetched. Client details include their system details in terms of software’s installed, services installed etc.

These details are generated in form of a report of specified format.

**Flexible Alerting engine** is designed for real life alerting with mail alerting types.  
SMTP mail with or without authentication and with fall-back to a 2nd will be sent to the user. It alerts the user about the system or about the web page on a regular time interval.

**2.2 PURPOSE**

The project **Server Monitoring** is aimed at developing an application that is able to test and verify whether the end users can interact with the website or web application.

* Monitoring different URLs as per end user’s requirements
* Assigning specific poll times to each URL being monitored
* Editing the existing URLs. (Disable/Deletion)
* Find the IP address of a given domain name
* Check the availability of port number for the given domain name
* Data is to be routed from its source to its destination
* Measures the amount and type of traffic on a particular network
* Fetches the details of the client in terms of software and services installed
* Maintaining log files for the same in order to store the status report
* Mail alerts are sent to the user

A network will be reported as unavailable if:

* Website is not accessible.
* Response time of the website crosses given poll time.
* HTTP Response code of the response is 4XX or 5XX.
* The username/password is invalid in the case of authenticated websites.
* Network traffic is heavy.
* Server cannot fetch the details of the client

**2.3 SCOPE**

The project **Server Monitoring** is aimed at developing an application that is able to test and verify different services of a server and ensure that all of them are being delivered to end user adequately. It is developed in order to help the businesses to ensure that their

network is live and responding. Monitoring a server means that the server owner always knows if one or all of his services go down.

Scope of this project includes:

* Ensure that websites are up and running at peak performance at all times
* Daily/weekly reports indicating web page downtime details
* Pick the right web host for your business
* Controlling the network traffic
* Data is routed from its source to its destination through a series of routers, and across multiple networks
* Enhancement in reputation through improvement in customer experience
* Minimize downtime thus any risk of direct loss of revenue to your business
* Helps user to find the IP address of a given website
* Allows users to find which ports are occupied on a system and which are free for the given domain name
* Fetch the client details and install programs to their systems from the servers
* Notifying the users by e-mail

[Network monitoring](http://en.wikipedia.org/wiki/Website_monitoring) allows interested parties to track the health of a website or web application. A software program can periodically check to see if a website is down, if there are broken links or if errors have occurred on particular pages. For example, a web developer who hosts and maintains a website for a customer may want to be notified instantly when the site goes or when one of the web applications returns an error.

IP Address Finder helps user to find the IP address of a given website. The user enters the URL and the IP address of the URL is provided by the IP Address Finder. Port Availability allows users to find which ports are occupied on a system and which are free for the given domain name. Data is routed from its source to its destination through a series of routers, and across multiple networks. At each stage, the IP address in the packet header is sufficient information to determine the next hop; no additional protocol headers are required.

Memory and Performance Management is the process of determining the network resources required to prevent a performance or availability impact on business-critical applications. System snapshot is developed to provide facility to their end users i.e. employees of making a software request if required. All the client’s details associated with a server can be fetched. Client details include their system details in terms of software installed, services installed etc. These details are generated in form of a report

of specified format. In the second module the same report is generated but giving current system details.

**2.4 OVERALL DESCRIPTION**

**2.4.1 Project features**

Some features of this system will be

* Monitoring different URLs as per end user’s requirements
* Find the IP address of a given domain name
* Fetch the client details and install programs to their systems from the servers
* Notifying the users by e-mail
* Maintaining log files for the same in order to store the status report
* Data is routed from its source to its destination through a series of routers, and across multiple networks
* The application will not stop until it is closed using the exit button. The application will continue to work if minimized or closed using “CLOSE” button.

Reports may be required to be generated in case of site failure.

It is developed in order to help the businesses to ensure that their website is live and responding. It also helps in reducing the downtime of websites and makes them always available to end-users. Apart from monitoring the status of the website, it also it also finds the IP address of a given domain name and checks the availability of the port for it.

**2.4.2 User classes and characteristics**

**Server Monitoring** is composed of four modules:

* Extensive Network Monitoring
* Memory and Performance Management
* System Snapshot
* Flexible Alerting Engine

Brief description on the modules:

1. **Extensive Network Monitoring** is aimed at developing an application that is able to test and verify whether the end users can interact with the website or web application. It is developed in order to help the businesses to ensure that their website is live and responding. It also helps in reducing the downtime of websites and makes them always available to end-users.

Apart from monitoring the status of the website, it also finds the IP address of a given domain name and checks the availability of the port for it. The IP Routing protocols enable routers to build up a forwarding table that correlates final destinations with next hop addresses.

1. **Memory and Performance Management** is the process of determining the network resources required to prevent a performance or availability impact on business-critical applications. Performance management is the practice of managing network service response time, consistency, and quality for individual and overall services.
2. **System Snapshots** is developed to provide facility to their end users i.e. employees of making a software request if required.

It is composed of two modules:

* 1. Fetching client system details on server
  2. Fetching current system details.

All the clients’ details associated with a server can be fetched. Client details include their system details in terms of software’s installed, services installed etc.

These details are generated in form of a report of specified format.

1. **Flexible Alerting Engine** uses log files, which are written with use of certain libraries provided in c#. The “log4net” namespace has been included for the purpose. Since mail needs to be sent therefore an additional functionality of SMTP settings is provided in the same. Here the settings can be edited as per user’s interest

**2.5 ASSUMPTIONS AND DEPENDENCIES**

**None**

**2.6 APPORTIONING OF REQUIREMENTS**

**Not required**

**2.7 SPECIFIC REQUIREMENTS**

This section contains the software requirements to a level of details sufficient to enable designers to design the system, and testers to test that system.

**2.7.1 Hardware Specification**

|  |  |  |
| --- | --- | --- |
| **HARDWARE REQUIREMENTS** | | |
| **PROCESSOR** | **RAM** | **DISK SPACE** |
| Pentium IV or higher | 512 MB or higher | 100 MB or higher if available |

**2.7.2 Software Specification**

|  |  |  |
| --- | --- | --- |
| **SOFTWARE REQUIREMENTS** | | |
| **OPERATING SYSTEM** | **DATABASE** | **PLATFORM** |
| Windows XP or any other higher version with Internet connection | SQL Server | Microsoft Visual Studio 2008 supporting C# applications |

**2.8 SOFTWARE SYSTEM ATTRIBUTES**

**2.8.1 SECURITY**

The database will be password protected. All details will be password protected. Administrator can view all details of the database in order to access the proceedings. The mail will be sent to the user in case of failure of the web link.

Application when closed will move to tray and keep the system active. It can only be shut through the file menu exit.

**2.8.2 MAINTAINABILITY**

The system will be designed in a maintainable manner. Log files are maintained for future use and maintenance. Log files are also written with use of certain libraries provided in c#. The “log4net” namespace has been included for the purpose. Since mail needs to be sent therefore an additional functionality of SMTP settings is provided in the same. Here the settings can be edited as per user’s interest.

**2.8.3 PORTABILITY**

This system will be easily portable on any windows based system that has MS Visual Studio, SQL server 2005 installed and can also be used with Oracle with slight modifications. It works under a good speed internet connection and a valid URL (syntax).

**SYSTEM DESIGN**

**3.1 DEFINITION**

**Designing is the most important phase of software development. It requires a careful planning and thinking on the part of the system designer. Designing software means to plan how the various parts of the software are going to achieve the desired goal. It should be done with utmost care because if the phase contains any error then that will affect the performance of the system, as a result it may take more processing time, more response time, extra coding workload etc.**

**Software design sits at the technical kernel of the software engineering process and is applied regardless of the software process model that is used. After the software requirements have been analyzed and specified, software design is the first of the three technical activities Designing, Coding and Testing that are required to build and verify the software. Each activity transforms information in such a manner that ultimately results in validated computer software.**

**3.2 DESIGN GOALS**

* **The following goals were kept in mind while designing the system:**
* **Make System User-Friendly so that system could be used efficiently and system could act as catalyst in achieving objectives**
* **Make System Compatible so that it should fit in the total integrated system. Future maintenance and enhancement must be less**
* **Make the System Compatible so that it could integrate other modules of system into itself**
* **Make the System Reliable, Understandable and Cost-Effective**

**During this entire iterative process the following principles continually guide the software Engineer.**

* **The design should be traceable to the Analysis Model.**
* **The design should “minimize the intellectual distance” between the software and the problem as it exists in the real world.**
* **The design should Exhibit Uniformity and Integrity.**
* **The design should be designed to accommodate changes.**
* **Design is not coding and coding is not design.**
* **The design should be reviewed to minimize conceptual (semantic) errors.**

**From the analysis, design requirements are formulated. The requirements for the new system are those features that must be incorporated to produce the improvements. These requirements are determined by comparing current performance with the objectives for acceptable systems performance.**

**The new system should have the following features:**

* **Greater speed of processing**
* **Effective procedure to eliminate errors**
* **Better accuracy**
* **Faster retrieval of information**
* **Integration of data**
* **Larger capacity of storing data with reduced cost**

**To achieve these features, several alternatives must be studied and evaluated. One alternative may not satisfy all the features. The analyst then selects those that are feasible economically, technically and operationally; the approach may emphasize the introduction of computerized system, replacement staff, changes in operating procedures, or a combination of several options.**

**The analyst often suggests inputs, process, and reporting and control procedures to help the management in decision making techniques. The procedures may be manual or automated but these will be useful in meeting systems requirements. Management will decide whether to accept and use them.**

**The role of a computer in a design revolves round its capabilities for calculation, storage and retrieval of data, summarization, sorting, classification and communication of data.**

**3.3 OUTPUT DESIGN**

In the output design, the emphasis is on producing a hard copy of the information requested or displaying the output on the CRT screen in a predetermined format. Two of the most output media today are printers and the screen.

Most users now access their reports from either a hard copy or screen display. Computer’s output is the most important and direct source of information to the user, efficient, logical, output design should improve the systems relations with the user and help in decision-making.

As the outputs are the most important source of information to the user, better design should improve the systems relations and also should help in decision-making. The output device’s capability, print quality, response time requirements etc should also be considered, form design elaborates the way the output is presented and layout available for capturing information. It’s very helpful to produce the clear, accurate and speedy information for end users.

**3.4 INPUT DESIGN**

In the input design, user-originated inputs are converted into a computer-based system format. It also includes determining the record media, method of input, speed of capture and entry on to the screen. Online data entry accepts commands and data through a keyboard. The major approach to input design is the menu and the prompt design. In each alternative, the user’s options are predefined. The data flow diagram indicates logical data flow, data stores, source and destination. Input data are collected and organized into a group of similar data once identified input media are selected for processing.

In this software, importance is given to develop Graphical User Interface (GUI), which is an important factor in developing efficient and user friendly software. For inputting user data, attractive forms are designed. User can also select the desired options from the menu, which provides all possible facilities. Also the important input format is designed

in such a way that accidental errors are avoided. The user has to input only just the minimum data required, which also helps in avoiding the errors that the users may make. Accurate designing of the input format is very important in developing efficient software. The goal of input design is to make entry as easy, logical and free from errors.

**3.5 Table Specifications**

Table 3.1: Table Specifications

|  |  |  |
| --- | --- | --- |
| **Table Name: URL Details** | | |
| **Fieldname** | **Data Type** | **Key** |
| SNo | Int | - |
| MonitorName | nchar(30) | - |
| URL | nchar(30) | Primary key |
| Username | nchar(30) | - |
| PassWord | nchar(30) | - |
| Email | nchar(30) | - |
| PollTime | int | - |
| IsActive | bit | - |
| CreatedOn | datetime | - |

**OVERVIEW OF LANGUAGE USED**

**4.1 About C# & MYSQL**

**C#**

C# (pronounced *see sharp*) is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within its [.NET](http://en.wikipedia.org/wiki/.NET_Framework)initiative and later approved as a standard by [Ecma](http://en.wikipedia.org/wiki/Ecma_International) (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure.

C# is intended to be a

* Simple
* Modern
* General-purpose
* Object Oriented programming language

Its development team is led by [Anders Hejlsberg](http://en.wikipedia.org/wiki/Anders_Hejlsberg). The most recent version is [C# 4.0](http://en.wikipedia.org/wiki/C_Sharp_4.0), which was released on April 12, 2010.

During the development of the .NET Framework, the [class libraries](http://en.wikipedia.org/wiki/Base_Class_Library) were originally written using a [managed code](http://en.wikipedia.org/wiki/Managed_code) compiler system called *Simple Managed C* (SMC). In January 1999, [Anders Hejlsberg](http://en.wikipedia.org/wiki/Anders_Hejlsberg) formed a team to build a new language at the time called Cool, which stood for "C-like Object Oriented Language". Microsoft had considered keeping the name "Cool" as the final name of the language, but chose not to do so for trademark reasons. By the time the .NET project was publicly announced at the July 2000 [Professional Developers Conference](http://en.wikipedia.org/wiki/Professional_Developers_Conference), the language had been renamed C#, and the class libraries and [ASP.NET](http://en.wikipedia.org/wiki/ASP.NET) runtime had been ported to C#.

C#'s principal designer and lead architect at Microsoft is Anders Hejlsberg, who was previously involved with the design of Pascal, Embarcadero (formerly CodeGear Delphi and Borland Delphi), and [Visual J++](http://en.wikipedia.org/wiki/Visual_J%2B%2B). In interviews and technical papers he has stated

that flaws in most major programming languages (e.g. [C++](http://en.wikipedia.org/wiki/C%2B%2B), [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), [Delphi](http://en.wikipedia.org/wiki/Embarcadero_Delphi), and [Smalltalk](http://en.wikipedia.org/wiki/Smalltalk)) drove the fundamentals of the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), which, in turn, drove the design of the C# language itself.

Since the release of C# 2.0 in November of 2005, the C# and Java languages have evolved on increasingly divergent trajectories, becoming somewhat less similar. One of the first major departures came with the addition of generics to both languages, with vastly different implementations. C# makes use of [reification](http://en.wikipedia.org/wiki/Reification_(computer_science)) to provide "first-class" generic objects that can be used like any other class, with code generation performed at class-load time. By contrast, Java's generics are essentially a language syntax feature, and they do not affect the generated byte code, because the compiler performs [type erasure](http://en.wikipedia.org/wiki/Type_erasure) on the generic type information after it has verified its correctness.

Furthermore, C# has added several major features to accommodate functional-style programming, culminating in their [LINQ](http://en.wikipedia.org/wiki/LINQ) extensions released with C# 3.0 and its supporting framework of [lambda expressions](http://en.wikipedia.org/wiki/Lambda_expressions), [extension methods](http://en.wikipedia.org/wiki/Extension_method), and anonymous classes. These features enable C# programmers to use functional programming techniques, such as [closures](http://en.wikipedia.org/wiki/Closure_(computer_science)), when it is advantageous to their application.

C# used to have a [mascot](http://web.archive.org/web/20060208020139/msdn.microsoft.com/vcsharp/art/csharp_team.gif) called Andy (named after Anders Hejlsberg). It was retired on 29 Jan 2004.

**Primary goals**

The ECMA standard lists these design goals for C#:

* C# language is intended to be a simple, modern, general-purpose, object-oriented programming language.
* The language, and implementations thereof, should provide support for software engineering principles such as [strong type checking](http://en.wikipedia.org/wiki/Strongly_typed_programming_language), [array bounds checking](http://en.wikipedia.org/wiki/Bounds_checking), detection of attempts to use uninitialized variables, and [automatic garbage collection](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). Software robustness, durability, and programmer productivity are important.
* The language is intended for use in developing [software components](http://en.wikipedia.org/wiki/Software_components) suitable for deployment in distributed environments.
* Source code portability is very important, as is programmer portability, especially for those programmers already familiar with C and C++.
* Support for [internationalization](http://en.wikipedia.org/wiki/Internationalization_and_localization) is very important.
* C# is intended to be suitable for writing applications for both hosted and [embedded systems](http://en.wikipedia.org/wiki/Embedded_system), ranging from the very large that use sophisticated [operating systems](http://en.wikipedia.org/wiki/Operating_system), down to the very small having dedicated functions.
* Although C# applications are intended to be economical with regard to memory and [processing power](http://en.wikipedia.org/wiki/Clock_rate) requirements, the language was not intended to compete directly on performance and size with C, C++ or assembly language.

**Evolution**

In the course of its development, the C# language has gone through several versions:

Table 4.1 Summary of Versions

**Summary of versions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **C# 2.0** | **C# 3.0** | **C# 4.0** | **C# 5.0 (planned)** |
| **Features added** | * Generics * Partial types * Anonymous methods * Iterators * Nullable types * Private setters (properties) * Method group conversions (delegates) | * Implicitly typed local variables * Object and collection initializers * Auto-Implemented properties * Anonymous types * Extension methods * Query expressions * Lambda expressions * Expression trees | * Dynamic binding * Named and optional arguments * Generic co- and contravariance | * Asynchronous methods * Compiler as a service |

The most recent version is [C# 4.0](http://en.wikipedia.org/wiki/C_Sharp_4.0), which was released on April 12, 2010.

**Difference between C# and C/C++ and JAVA**

* Some notable features of C# that distinguish it from C and C++ (and Java, where noted) are:
* It has no global variables or functions. All methods and members must be declared within classes. Static members of public classes can substitute for global variables and functions.
* Local variables cannot shadow variables of the enclosing block, unlike C and C++. [Variable shadowing](http://en.wikipedia.org/wiki/Variable_shadowing) is often considered confusing by C++ texts.
* C# supports a strict [Boolean data type](http://en.wikipedia.org/wiki/Boolean_data_type), bool. Statements that take conditions, such as **while** and **if**, require an expression of a type that implements the **true** operator, such as the boolean type. While C++ also has a boolean type, it can be freely converted to and from integers, and expressions such as **if**(a) require only that a is convertible to bool, allowing a to be an int, or a pointer. C# disallows this "integer meaning true or false" approach, on the grounds that forcing programmers to use expressions that return exactly bool can prevent certain types of common programming mistakes in C or C++ such as **if** (a = b) (use of assignment = instead of equality ==).
* In C#, memory address pointers can only be used within blocks specifically marked as *unsafe*, and programs with unsafe code need appropriate permissions to run.
* Managed memory cannot be explicitly freed; instead, it is automatically garbage collected. Garbage collection addresses the problem of [memory leaks](http://en.wikipedia.org/wiki/Memory_leak) by freeing the programmer of responsibility for releasing memory that is no longer needed.
* In addition to the **try**...**catch** construct to handle exceptions, C# has a **try**...**finally** construct to guarantee execution of the code in the **finally** block.
* [Multiple inheritances](http://en.wikipedia.org/wiki/Multiple_inheritance) is not supported, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication and simplify architectural requirements throughout CLI.
* C#, like C++, but unlike Java, supports [operator overloading](http://en.wikipedia.org/wiki/Operator_overloading).
* C# is more [type safe](http://en.wikipedia.org/wiki/Type_safety) than C++. The only implicit conversions by default are those that are considered safe, such as widening of integers.
* [Enumeration](http://en.wikipedia.org/wiki/Enumerated_type) members are placed in their own [scope](http://en.wikipedia.org/wiki/Scope_(programming)).
* C# provides [properties](http://en.wikipedia.org/wiki/Property_(programming)) as [syntactic sugar](http://en.wikipedia.org/wiki/Syntactic_sugar) for a common pattern in which a pair of methods, [accessor (getter) and mutator (setter)](http://en.wikipedia.org/wiki/Mutator_method)encapsulate operations on a single [attribute](http://en.wikipedia.org/wiki/Attribute_(computing)) of a class.
* Full type [reflection](http://en.wikipedia.org/wiki/Reflection_(computer_science)) and discovery is available.

**Common Type System (CTS)**

C# has a *unified type system*. This unified type system is called [Common Type System](http://en.wikipedia.org/wiki/Common_Type_System) (CTS). A unified type system implies that all types, including primitives such as integers, are subclasses of the System.Object class. For example, every type inherits a ToString() method.

CTS separate data types into two categories.

* Value types
* Reference types

Instances of value types do not have referential identity or referential comparison semantics - equality and inequality comparisons for value types compare the actual data values within the instances, unless the corresponding operators are overloaded. Value types are derived fromSystem.ValueType, always have a default value, and can always be created and copied. Some other limitations on value types are that they cannot derive from each other (but can implement interfaces) and cannot have an explicit default (parameter less) constructor. Examples of value types are all primitive types, such as int (a signed 32-bit integer), float (a 32-bit IEEE floating-point number), char (a 16-bit Unicode code unit), and System.DateTime (identifies a specific point in time with nanosecond precision). Other examples are enum(enumerations) and struct (user defined structures).

In contrast, reference types have the notion of referential identity - each instance of a reference type is inherently distinct from every other instance, even if the data within both instances is the same. This is reflected in default equality and inequality comparisons for reference types, which test for referential rather than structural equality, unless the corresponding operators are overloaded (such as the case forSystem.String).

In general, it is not always possible to create an instance of a reference type, nor to copy an existing instance, or perform a value comparison on two existing instances, though specific reference types can provide such services by exposing a public constructor or implementing a corresponding interface (such as ICloneable or IComparable). Examples of reference types are object (the ultimate base class for all other C# classes), System.String (a string of Unicode characters), and System.Array (a base class for all C# arrays). Both type categories are extensible with user-defined types.

**Functions of CTS**

* To establish a framework that helps enable cross-language integration, type safety, and high performance code execution.
* To provide an [object-oriented](http://en.wikipedia.org/wiki/Object-oriented) model that supports the complete implementation of many programming languages.
* To define rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.
* The CTS also defines the rules that ensure that the data types of objects written in various languages are able to interact with each other.
* The CTS also specifies the rules for type visibility and access to the members of a type, i.e. the CTS establishes the rules by which assemblies form scope for a type, and the Common Language Runtime enforces the visibility rules.
* The CTS defines the rules governing [type inheritance](http://en.wikipedia.org/wiki/Type_inheritance), virtual methods and object lifetime.
* Languages supported by .NET can implement all or some common data types

**Boxing and Unboxing**

*Boxing* is the operation of converting a value of a value type into a value of a corresponding reference type. Boxing in C# is implicit.

*Unboxing* is the operation of converting a value of a reference type (previously boxed) into a value of a value type. Unboxing in C# requires an explicit [type cast](http://en.wikipedia.org/wiki/Type_conversion). A boxed object of type T can only be unboxed to a T (or a null able T).

**Generics**

[Generics](http://en.wikipedia.org/wiki/Generic_programming) were added to version 2.0 of the C# language. Generics use type parameters, which make it possible to design classes and methods that do not specify the type used until the class or method is instantiated. The main advantage is that one can use generic type parameters to create classes and methods that can be used without incurring the cost of runtime casts or boxing operations.

**Preprocessor**

C# features "preprocessor directives" (though it does not have an actual preprocessor) based on the [C preprocessor](http://en.wikipedia.org/wiki/C_preprocessor) that allow programmers to define [symbols](http://en.wikipedia.org/wiki/Symbols), but not macros. Conditionals such as #if, #endif, and #else are also provided. Directives such as#region give hints to editors for [code folding](http://en.wikipedia.org/wiki/Code_folding).

**4.2 SQL**

SQL (often referred to as Structured Query Language) is a [programming language](http://en.wikipedia.org/wiki/Programming_language) designed for managing data in [relational database management systems](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS).

Originally based upon [relational algebra](http://en.wikipedia.org/wiki/Relational_algebra) and [tuple relational calculus](http://en.wikipedia.org/wiki/Tuple_relational_calculus), its scope includes data insert, query, [update and delete](http://en.wikipedia.org/wiki/Data_Manipulation_Language), [schema](http://en.wikipedia.org/wiki/Database_schema) creation and modification, and data access control.

SQL was one of the first commercial languages for [Edgar F. Codd](http://en.wikipedia.org/wiki/Edgar_F._Codd)'s [relational model](http://en.wikipedia.org/wiki/Relational_model), as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not adhering to [the relational model as described by Codd](http://en.wikipedia.org/wiki/Codd%27s_12_rules), it became the most widely used database language. Though often described as, and to a great extent is a declarative, SQL also includes [procedural](http://en.wikipedia.org/wiki/Procedural_programming) elements. SQL became a [standard](http://en.wikipedia.org/wiki/Technical_standard) of the American (ANSI) in 1986, and of the [International Organization for Standards](http://en.wikipedia.org/wiki/International_Organization_for_Standards) (ISO) in 1987. Since then the standard has been enhanced several times with added features. However, issues of SQL code portability between major RDBMS products still exist due to lack of full compliance with, or different interpretations of the standard. Among the reasons mentioned are the large size, and incomplete specification of the standard, as well as [vendor lock-in](http://en.wikipedia.org/wiki/Vendor_lock-in).

The SQL language is sub-divided into several language elements, including:

* **Clauses,** which are constituent components of statements and queries. (In some cases, these are optional.)
* **Expressions**, which can produce either [scalar](http://en.wikipedia.org/wiki/Scalar_(computing)) values or [tables](http://en.wikipedia.org/wiki/Table_(database)) consisting of [columns](http://en.wikipedia.org/wiki/Column_(database)) and [rows](http://en.wikipedia.org/wiki/Row_(database)) of data.
* **Predicates**, which specify conditions that can be evaluated to SQL [three-valued logic (3VL)](http://en.wikipedia.org/wiki/Ternary_logic) or [Boolean](http://en.wikipedia.org/wiki/Boolean_logic) (true/false/unknown) [truth values](http://en.wikipedia.org/wiki/Truth_value) and which are used to limit the effects of statements and queries, or to change program flow.
* **Queries**, which retrieve the data based on specific criteria. This is the most important element of SQL.
* **Statements**, which may have a persistent effect on schemata and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
* SQL statements also include the [semicolon](http://en.wikipedia.org/wiki/Semicolon) (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
* [**Insignificant whitespace**](http://en.wikipedia.org/wiki/Whitespace_(computer_science)) is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

**Queries**

The most common operation in SQL is the query, which is performed with the declarative [SELECT](http://en.wikipedia.org/wiki/Select_(SQL)) statement. SELECT retrieves data from one or more [tables](http://en.wikipedia.org/wiki/Table_(database)), or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO syntax that exists in some databases.

Queries allow the user to describe desired data, leaving the [database management system (DBMS)](http://en.wikipedia.org/wiki/Database_management_system) responsible for [planning](http://en.wikipedia.org/wiki/Query_plan), [optimizing](http://en.wikipedia.org/wiki/Query_optimizer), and performing the physical operations necessary to produce that result as it chooses.

* A query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk ("\*") can also be used to specify that the query should return all columns of the queried tables. SELECT is the most complex statement in SQL, with optional keywords and clauses.
* The [FROM](http://en.wikipedia.org/wiki/From_(SQL)) clause which indicates the table(s) from which data is to be retrieved. The FROM clause can include optional [JOIN](http://en.wikipedia.org/wiki/Join_(SQL)) subclauses to specify the rules for joining tables.
* The [WHERE](http://en.wikipedia.org/wiki/Where_(SQL)) clause includes a comparison predicate, which restricts the rows returned by the query. The WHERE clause eliminates all rows from the result set for which the comparison predicate does not evaluate to True.
* The GROUP BY clause is used to project rows having common values into a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The WHERE clause is applied before the GROUP BY clause.
* The [HAVING](http://en.wikipedia.org/wiki/Having_(SQL)) clause includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.
* The [ORDER BY](http://en.wikipedia.org/wiki/Order_by_(SQL)) clause identifies which columns are used to sort the resulting data, and in which direction they should be sorted (options are ascending or descending). Without an ORDER BY clause, the order of rows returned by an SQL query is undefined.

**4.3 SQL SERVER**

Microsoft SQL Server is a [relational database server](http://en.wikipedia.org/wiki/Relational_database_management_system), developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft): It is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for different workloads (ranging from small applications that store and retrieve data on the same computer, to millions of users and computers that access huge amounts of data from the Internet at the same time).True to its namesake, Microsoft SQL Server's primary [query languages](http://en.wikipedia.org/wiki/Query_language) are [T-SQL](http://en.wikipedia.org/wiki/Transact-SQL) and [ANSI SQL](http://en.wikipedia.org/wiki/SQL).

**ARCHITECTURE**

Protocol layer

Protocol layer implements the external interface to SQL Server. All operations that can be invoked on SQL Server are communicated to it via a Microsoft-defined format, called [Tabular Data Stream](http://en.wikipedia.org/wiki/Tabular_Data_Stream) (TDS). TDS is an application layer protocol, used to transfer data between a database server and a client. Initially designed and developed by Sybase Inc. for their [Sybase SQL Server](http://en.wikipedia.org/wiki/Sybase_SQL_Server) relational database engine in 1984, and later by Microsoft in Microsoft SQL Server, TDS packets can be encased in other physical transport dependent protocols, including [TCP/IP](http://en.wikipedia.org/wiki/TCP/IP), [Named pipes](http://en.wikipedia.org/wiki/Named_pipe), and [Shared memory](http://en.wikipedia.org/wiki/Shared_memory). Consequently, access to SQL Server is available over these protocols. In addition, the SQL Server API is also exposed over [web services](http://en.wikipedia.org/wiki/Web_service).

**DATA STORAGE**

The main unit of [data storage](http://en.wikipedia.org/wiki/Computer_data_storage) is a [database](http://en.wikipedia.org/wiki/Database), which is a collection of tables with [typed](http://en.wikipedia.org/wiki/Type_system) columns. SQL Server supports different data types, including [primary types](http://en.wikipedia.org/wiki/Primary_type) such as *Integer*, *Float*, *Decimal*, *Char* (including character strings), *Varchar* (variable length character strings), binary (for unstructured [blobs](http://en.wikipedia.org/wiki/Binary_large_object) of data), *Text* (for textual data) among others. The [rounding](http://en.wikipedia.org/wiki/Rounding) of floats to integers uses either

Symmetric Arithmetic Rounding or Symmetric Round Down (*Fix*) depending on arguments

Buffer management

SQL Server [buffers](http://en.wikipedia.org/wiki/Data_buffer) pages in [RAM](http://en.wikipedia.org/wiki/RAM) to minimize disc I/O. Any 8 KB page can be buffered in-memory, and the set of all pages currently buffered is called the buffer cache. The amount of memory available to SQL Server decides how many pages will be cached in memory. The buffer cache is managed by the *Buffer Manager*. Either reading from or writing to any page copies it to the buffer cache. Subsequent reads or writes are redirected to the in-memory copy, rather than the on-disc version. The page is updated on the disc by the Buffer Manager only if the in-memory cache has not been referenced for some time. While writing pages back to disc, asynchronous I/O is used whereby the I/O operation is done in a background thread so that other operations do not have to wait for the I/O operation to complete. Each page is written along with its checksum when it is written. When reading the page back, its checksum is computed again and matched with the stored version to ensure the page has not been damaged or tampered with in the meantime.

Concurrency and locking

SQL Server allows multiple clients to use the same database concurrently. As such, it needs to control concurrent access to shared data, to ensure data integrity - when multiple clients update the same data, or clients attempt to read data that is in the process of being changed by another client. SQL Server provides two modes of concurrency control: pessimistic concurrency and optimistic concurrency.

When pessimistic concurrency control is being used, SQL Server controls concurrent access by using locks. Locks can be either shared or exclusive. Exclusive lock grants the user exclusive access to the data - no other user can access the data as long as the lock is held. Shared locks are used when some data is being read - multiple users can read from data locked with a shared lock, but not acquire an exclusive lock. The latter would have to wait for all shared locks to be released. Locks can be applied on different levels of

granularity - on entire tables, pages, or even on a per-row basis on tables. For indexes, it can either be on the entire index or on index leaves.

The level of granularity to be used is defined on a per-database basis by the database administrator. While a fine grained locking system allows more users to use the table or index simultaneously, it requires more resources. So it does not automatically turn into higher performing solution. SQL Server also includes two more lightweight mutual exclusion solutions - latches and spinlocks - which are less robust than locks but are less resource intensive. SQL Server uses them for DMVs and other resources that are usually not busy. SQL Server also monitors all worker threads that acquire locks to ensure that they do not end up in deadlocks - in case they do, SQL Server takes remedial measures, which in many cases is to kill one of the threads entangled in a deadlock and rollback the transaction it started.

To implement locking, SQL Server contains the *Lock Manager*. The Lock Manager maintains an in-memory table that manages the database objects and locks, if any, on them along with other metadata about the lock. Access to any shared object is mediated by the lock manager, which either grants access to the resource or blocks it.

**PROGRAMMABILITY**

T-SQL

T-SQL (Transact-SQL) is the primary means of programming and managing SQL Server. It exposes keywords for the operations that can be performed on SQL Server, including creating and altering database schemas, entering and editing data in the database as well as monitoring and managing the server itself. Client applications that consume data or manage the server will leverage SQL Server functionality by sending T-SQL queries and statements which are then processed by the server and results (or errors) returned to the client application. SQL Server allows it to be managed using T-SQL. For this it exposes read-only tables from which server statistics can be read. Management functionality is exposed via system-defined stored procedures which can be invoked from T-SQL queries to perform the management operation. It is also possible

to create linked Server using T-SQL. Linked server allows operation to multiple servers as one query.

SQL Native Client

SQL Native Client is the native client side data access library for Microsoft SQL Server, version 2005 onwards. It natively implements support for the SQL Server features including the Tabular Data Stream implementation, support for mirrored SQL Server databases, full support for all data types supported by SQL Server, asynchronous operations, query notifications, encryption support, as well as receiving multiple result sets in a single database session. SQL Native Client is used under the hood by SQL Server plug-ins for other data access technologies, including ADO or OLE DB. The SQL Native Client can also be directly used, bypassing the generic data access layers.

**CONCLUSION**

The project is an application that is able to test and verify different services of a server and ensure that all of them are being delivered to end user adequately. It is developed in order to help the businesses to ensure that their network is live and responding. Monitoring a server means that the server owner always knows if one or all of his services go down.

IP Address Finder helps user to find the IP address of a given website. The user enters the URL and the IP address of the URL is provided by the IP Address Finder. IP address of any entered valid Domain Name is given. Port Availability allows users to find which ports are occupied on a system and which are free for the given domain name. Any domain name and its port are entered and its availability is checked on the system and result is displayed.

**System Snapshot** helps us to fetch the system details of the client and the server. All the client’s details associated with a server can be fetched. Client details include their system details in terms of software’s installed, services installed etc. These details are generated in form of a report of specified format.

**Flexible Alerting Engine** will be designed for real life alerting with mail alerting types. **SMTP mail** with or without authentication and with fall-back to a 2ndare sent to the end user. It alerts the user about the system or about the web page on a regular time interval.

The project could very well be enhanced further as per the requirements and help in enhancing the network’s performance.

**FUTURE ENHANCEMENT**

**Since there is always possibility of making improvements to a system, in order to get better service out of it. Some areas of future extensibility are also possible in the system information utility and software requisition, which are as follow:**

* More components can be monitored.
* SMS in addition to Mails can be sent to the authorities for quick action and repair of the link.
* The User Interface can be made more user friendly and attractive
* Graphs and line charts can be added for quick view of the monitoring report of a specific link
* Location of a given IP address can be traced using Google Maps and location facilities and enhancing the database
* More clients can be added to the system

**REFERENCES**

1. Belmaks Solution Content Team, Dreamtech Software Development Team, ASP.NET 2.0 Black Book, Paraglyph Press Publishers, Edition: 2007-08, Ch 4-8, ISBN : 9781933097176
2. Jon Skeet, C# in depth, Manning Publications, Pap/Psc Edition, Ch 1-5, ISBN: 1935182471
3. SAMS, *SAMS Teach Yourself C# in 21 day,* 2010-11edition, Ch- 6-11, ISBN: 0672327112
4. Techotopia, C# Essentials,2nd Edition, O'Reilly Media, Ch-3-6,

ISBN: 9780596003159

1. http://www.xdevsoftware.com/blog/post/Website-Monitoring-Simple-Website-Monitor.aspx
2. http://www.eggheadcafe.com/articles/20040516.asp
3. http://www.howtogeek.com/howto/programming/get-ip-address-from-dns-hostname-in-c/
4. http://stackoverflow.com/questions/570098/in-c-how-to-check-if-a-tcp-port-is-available
5. http://www.csharphelp.com/2006/01/access-registry-using-c/

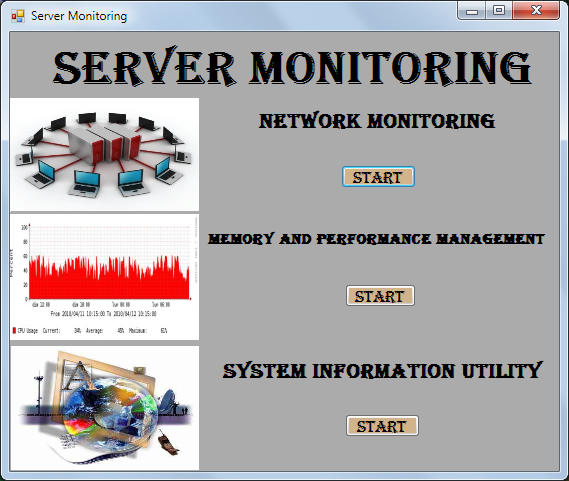
**APPENDIX-A**

**A.1 SNAPSHOTS**

**SERVER MONITORING**

Home page

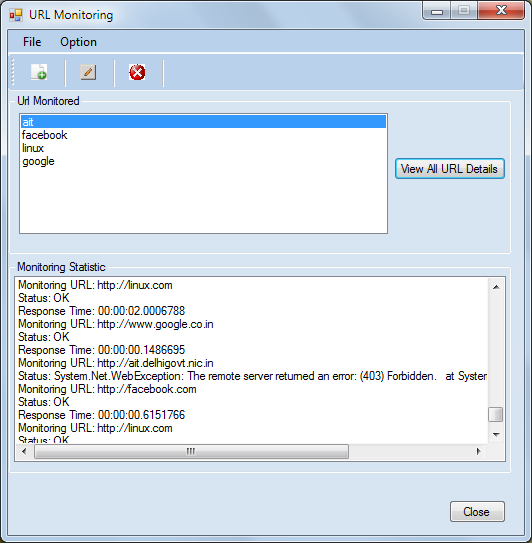
It is the home page of the project. It displays all the modules present in the project.



**NETWORK MONITORING**

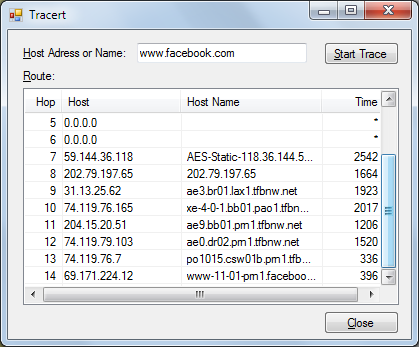
URL Monitoring Screen

This screen provides the user with different monitors and their respective statistics.



Route Tracing

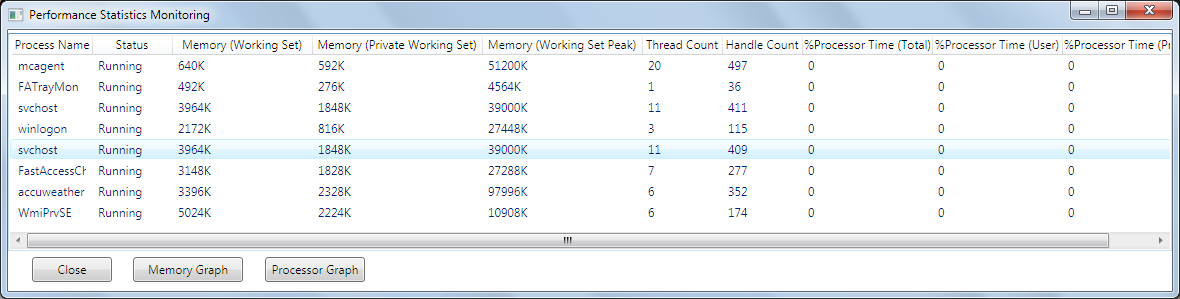
This is used to trace the route of a given URL to its destination. It enables routers to build up a forwarding table that correlates final destinations with next hop addresses. At each stage, the IP address in the packet header is sufficient information to determine the next hop.



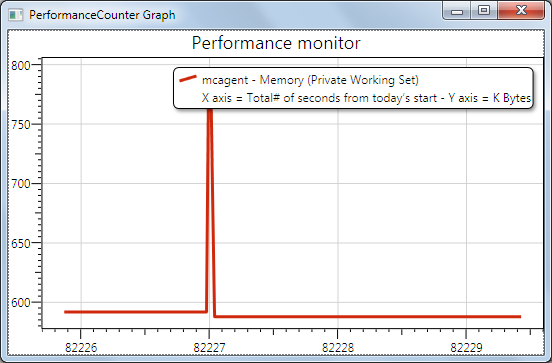
**MEMORY AND PERFORMANCE MANAGEMENT**

Performance Monitoring Screen

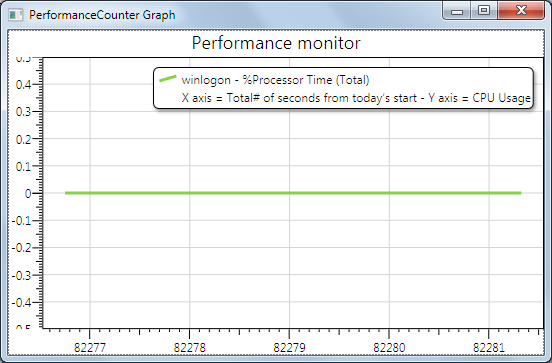
It displays the performance of the system in graphical format as well.

****

Performance graph for memory management

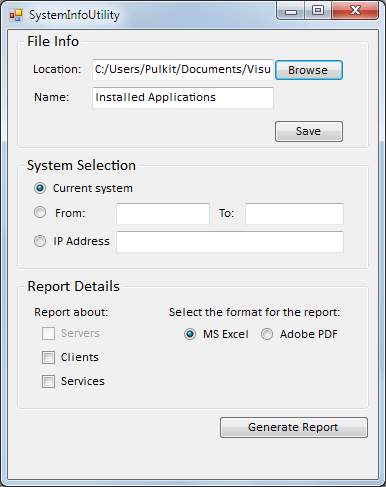
****

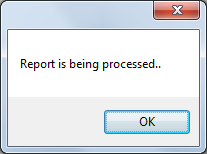
Performance graph for memory management

****

**SYSTEM INFORMATION UTILITY**

Processing of Report





**APPENDIX-B**

**B.1 SOURCE CODE**

**Home page**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using MemoryPerformanceMonitoring;

namespace UserInterface

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

Form11 window = new Form11();

window.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Window1 window = new Window1();

window.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

Form12 window = new Form12();

window.ShowDialog();

//Application.Run(new Form12());

}

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

{

}

}

}

**Network monitoring form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace UserInterface

{

public partial class Form11 : Form

{

public Form11()

{

InitializeComponent();

}

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

{

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

//Application.Run(new urlmonitoringform());

urlmonitoringform window= new urlmonitoringform();

//window.Text = "a";

window.Show();

}

private void linkLabel2\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Form2 window = new Form2();

window.Show();

}

private void linkLabel3\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Form3 window = new Form3();

window.Show();

}

private void linkLabel4\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

MainForm window = new MainForm();

window.Show();

//SMTP\_Settings window = new SMTP\_Settings();

//window.Show();

}

private void linkLabel1\_LinkClicked\_1(object sender, LinkLabelLinkClickedEventArgs e)

{

//Form13 window = new Form13();

//window.Show();

}

}

}

**System info utility**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Collections;

using Microsoft.Win32;

using System.Configuration;

using System.IO;

using Microsoft.Reporting.WinForms;

using System.ServiceProcess;

namespace UserInterface

{

public partial class Form12 : Form

{ List<Info> Systeminfo = new List<Info>();

string folderName;

string choice = "Current system";

ArrayList server = new ArrayList();

ArrayList client = new ArrayList();

public Form12()

{

InitializeComponent();

}

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

{

}

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

{

DialogResult result = folderBrowserDialog1.ShowDialog();

if (result == DialogResult.OK)

{

folderName = folderBrowserDialog1.SelectedPath;

this.txtReportloc.Text = folderName;

}

else

{

this.txtReportloc.Text = ConfigurationSettings.AppSettings["FILEPATH"].ToString();

}

}

if (this.chkboxClient.Checked)

{

GetSoftwareInstalled(networkComputers,"client");

}

if (this.chkboxServices.Checked)

{

GetServicesInstalled(networkComputers);

}

public ArrayList GetIPAddressRange()

{

string strtrange = startrange.Text;

string edrange = endrange.Text;

ArrayList arraylist = new ArrayList();

try

{

int StartIP = strtrange.LastIndexOf('.');

string IPSeries = strtrange.Remove(StartIP + 1);

int IPStart = Convert.ToInt32(strtrange.Remove(0, StartIP + 1));

int indexEndIP = edrange.LastIndexOf('.');

int IPEnd = Convert.ToInt32(edrange.Remove(0, indexEndIP + 1));

for (int tempIP = StartIP; tempIP <= IPEnd; tempIP++)

{

string IPt = IPSeries + tempIP.ToString();

arraylist.Add(IPt);

//Console.WriteLine(IP);

}

return arraylist;

}

catch (Exception ex)

{

return null;

} }

public static byte[] GetReport(ReportViewer reportViewer,string form)

{

try

{

Warning[] warnings;

string[] streamids;

string mimeType;

string encoding;

string extension;

byte[] bytes = null;

bytes = reportViewer.LocalReport.Render(form, null, out mimeType, out encoding, out extension, out streamids, out warnings);

return bytes;

}

catch (Exception e)

{

MessageBox.Show(e.Message);

return null;

}

}

}

}

**Memory performance module**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Collections.ObjectModel;

using System.Diagnostics;

using System.Threading;

namespace MemoryPerformanceMonitoring

{

/// <summary>

/// Interaction logic for Window1.xaml

/// </summary>

public partial class Window1 : Window

{

string strn = null;

Process[] proc = Process.GetProcesses();

//string[] arr= new string(

//instance name for which you want to monitor

// string sProcName = "regedit";

ObservableCollection<StatisticsData> \_StatisticsCollection =

new ObservableCollection<StatisticsData>();

public ObservableCollection<StatisticsData> StatisticsCollection

{ get { return \_StatisticsCollection; } }

public Window1()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

Thread MonitorThd = new Thread(this.StatisticsMonitoringThread);

MonitorThd.Start();

//MemoryandProcessCheck();

}

public void StatisticsMonitoringThread(object obj)

{

foreach (Process proce in proc)

{

//strn = proce.ProcessName;

MonitorMemoryandProcess(proce);

}

}

private Process CurrentlyRunning()

{

//get a list of all running processes on current system

// Process[] Processes = Process.GetProcesses();

//Iterate to every process to check if it is out required process

foreach (Process SingleProcess in proc)

{

//str = SingleProcess.ProcessName;

return SingleProcess;

}

return null;

}

private bool MonitorMemoryandProcess(Process po)

{

string ProcessStatus = null;

string[] str = new string[10];

LineGraph chart = plotter.AddLineGraph(ds, 3.0, String.Format("{0} - {1}", instanceName, strDisplayName));

if(strDisplayName.Contains("Memory"))

{

ds.SetYMapping(pi => pi.Value/1024);

plotter.AddLineGraph(ds, cr, 3.0, String.Format("{0} - {1}", "X axis = Total# of seconds from today’s start", "Y axis = K Bytes"));

}

else

{

ds.SetYMapping(pi => pi.Value);

plotter.AddLineGraph(ds, cr, 3.0, String.Format("{0} - {1}", "X axis = Total# of seconds from today’s start", "Y axis = CPU Usage"));

}

return chart;

}

private void chart\_DataChanged(object sender, EventArgs e)

{

LineGraph graph = (LineGraph)sender;

double mbytes = graph.DataSource.GetPoints().LastOrDefault().Y;

graph.Description = new PenDescription(String.Format("Memory - available {0} MBytes", mbytes));

}

}}