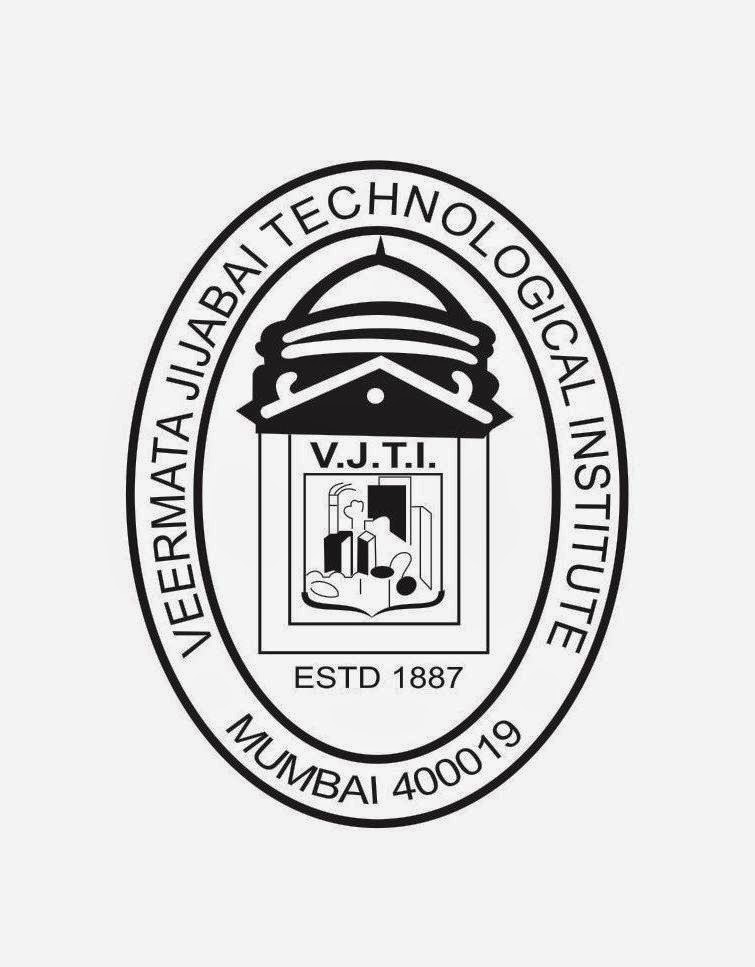
**NETWORK USABILITY MONITOR**



Under the guidance of

**Prof. P. M. Chawan**

**CERTIFICATE**

**STATEMENT BY CANDIDATE**

**TABLE OF CONTENTS**

1. Problem statement
   1. Introduction
   2. Background
      1. Previous attempt
      2. Current scenario
   3. Elaboration
2. Scope
   1. Features
   2. Constraints
   3. Performance requirements
      1. Response time
      2. Workload
      3. Scalability
      4. Platform
3. Resources
   1. Human resources
   2. Reusable software resources

**PROBLEM STATEMENT**

* 1. **Introduction:**

To design a network monitoring system

* 1. **Background:**
     1. **Previous attempt:**

It would be indecorous to state that the previous attempts made to design an all in one network monitor were fruitful, as they have found little practical usage today. They are mostly used by service providers, companies owning huge networks, etc. and find no use in local and small networks.

* + 1. **Current scenario:**

The existing network monitoring softwares are too complex and difficult to use for people from a non-networking or a non-computer background. These softwares use many technical terms which are jargon for computer and networking scientists. Several ways to measure network performance have been developed so far; however, performance management is still difficult because of lack of effective tools to evaluate network system usability. Also, the networking commands used in a Windows or Linux terminal have numerous uses and their output is static, i.e. we need to run them continuously for a time dependent analysis. We can state the inadequacies of this system as follows: -

1. Difficulty in understanding technical jargon for people from a non-networking background
2. Static outputs from Windows Command Prompt or Linux Terminal which compel users to continuously run them in order to view the change in statistics with respect to time
3. Lack of effective tools to evaluate network system usability
4. Problems are faced by the network users and administrators to determine as to what exactly is the fault and the way it should be dealt with; as an example, if a network gives warns us about limited access, the DNS doesn’t reply the host with the IP address and hence, the communication breaks, however, the actual reason behind the problem is often ignored.
   1. **Elaboration:**

Our aim is to design a network monitoring tool for any client system which can be useful in any environment by a network administrator; be it home, office, an educational institute or a computer laboratory. The goal is to develop a new performance evaluation tool for network system usability. In our approach, we measure network system usability and performance through the mechanisms installed in the clients. The purpose of the system is to make clear the behaviour of the client applications which allows us to measure system performance for client usability and perform analytics on the data, thus generating useful outputs for the user. Some of the aspects of our implementation are: -

1. Obtaining crucial data for the network and its parameters such as the access delay, processing time, etc.
2. The tool would run on a Windows or a Linux operating system on any simple PC.
3. The software would be standalone and freely distributable, which would mean that it could be shared on other devices without any chaotic methods.
4. The data would be represented graphically which would provide an attractive GUI for users.

**PROJECT SCOPE**

**Features:** This particular project is expected to have the following features:

* An easy to use graphical user interface (GUI) for system administrators. The system administrator won’t be required to type in commands in the terminal directly. Our easy-to-use GUI will provide an excellent interface using which a person need only click on particular buttons and he/she will receive all the necessary information in an easy to interpret manner.
* Real time graph based monitoring of network resources to help them identify any bottlenecks in the network. This will be accomplished by continuous packet transfer in the background. Based on the received data, graphs will plot to provide the necessary analytics of the network.
* Efficient tools to evaluate network system usability. Network commands to be executed indirectly in the terminal through our application, providing a layer of abstraction from background details.
* Easy detection of network problems
* Notifications to be sent to the system administrators through emails or text messages about any network issues
* Effective solutions to solve the identified problems

**Constraints:** Although the project will be advancement on the previous attempts, there are some limitations of the application. They are mentioned below:

* Notifications about network problems cannot be sent via text messages if Telnet protocol is not functional or is not installed.
* Performance enhancement can be achieved only on certain network configurations.
* Solutions to problems might require restarting the entire system, which is real time problem solutions are difficult to achieve.
* Only the total data flow in the network can be monitored and not the data flow through each node.
* Bandwidth of the network will affect the performance of the application.
* Network congestion might result in delayed monitoring and analysis of the usability features.

**Performance requirements:** Following are the four categories of performance measures of our application:

* Response Time: It is the amount of time required to load the application completely. This involves continuous packet transfer in the background to enable real time monitoring and graph based analysis. As such, a response time of about 2-3 seconds can be estimated under minimal requirement conditions. Variations in the response time can be caused due to heavy server payload in times of peak usage. A variation of 1-2 seconds is expected under such conditions. If there is no network connection, then the application will not start.
* Workload:

|  |  |  |
| --- | --- | --- |
| Scenario | Daily total | Time (in seconds) |
| Authentication | 200 | 5 |
| Graph analysis | 170 | 5,10,15(based on options) |
| Delay calculation | 150 | Varying |
| Notification system | 10 | 2 |
| Performance management | 120 | 10 |

* Scalability:

This application is meant to be used for large networks as well as small private networks. Ideal usage is in a college local area network (LAN) or other such institutions. As distance between nodes and server will affect the performance of the application, huge networks viz. wide area networks (WAN) or metropolitan area networks (MAN) might work with reduced performance and efficiency. Also, the bandwidth of the network will affect the performance.

The application will work with about 200 active nodes in the network and active servers. A peak usage of 400 active nodes can also be dealt with.

* Platform:

The application will be developed in Java. Therefore it is platform-independent. Although, the minimum requirements of 10MB hard disk space to store the application and 512MB of RAM for its execution. Software requirements include the pre-installation of Java Runtime Environment (JRE) of any current available version present on the Oracle website. A working network connection is required and a valid authentication for the system administrator.

**RESOURCES**

A part of software planning task is estimation of the resources required to accomplish the software development effort.

Following are the resources required in software development:-

1. **Human Resources**

**Developers**

A developer is the person who researches, designs, develops and tests the software. A software developer also takes part in developing the software UI and programming the software logic.

In this project, a group of 5 members (Vishal, Varun, Saif, Rohit, Dharmin) are the developers performing the above tasks.

* 1. **Reusable Software Resources**

Component-based software engineering (CBSE) emphasizes reusability—that is, the creation and reuse of software building blocks. Such building blocks, often called components, must be catalogued for easy reference, standardized for easy application, and validated for easy integration.

**Off-the-shelf components**

Existing software that can be acquired from a third party or that has been developed internally for a past project. COTS (commercial off-the-shelf) components are purchased from a third party, are ready for use on the current project, and have been fully validated.

**Full-experience components**

Existing specifications, designs, code, or test data developed for past projects that are similar to the software to be built for the current project. Members of the current software team have had full experience in the application area represented by these components. Therefore, modifications required for full-experience components will be relatively low-risk.

**Partial-experience components**

Existing specifications, designs, code, or test data developed for past projects that are related to the software to be built for the current project but will require substantial modification. Members of the current software team have only limited experience in the application area represented by these components. Therefore, modifications required for partial-experience components have a fair degree of risk.

**New components**

Software components that must be built by the software team specifically for the needs of the current project.

* 1. **System Requirements**

**Development Environment**

The environment that supports the software project, often called the software engineering environment (SEE), incorporates hardware and software. Hardware provides a platform that supports the tools (software) required to produce the work products that are an outcome of good software engineering practice.

**Hardware Requirements:**

* Random access memory: 512 MB
* Hard disk space: 10 MB
* At least one active network connection

**Software Requirements:**

* Operating System: Windows XP/Vista/7/8/8.1/10, Linux
* Administrative rights to access network parameters
* Pre-installed Java Runtime Environment