KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science and Engineering

An Investigation of Taibah University Network Application Faults Using a Monitoring Algorithm

A Project Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Science in Computer Engineering

Submitted By

Ahmed Elegl **3700468**
Mohammad Ammar **3704975**

Supervised by
Dr. Wael Alheadary

2020/2021

ABSTRACT

Network monitoring tools are significant tools for organizations since it provides stability to network and protection from downtime. In particular, the Network Management System (NMS) used by Taibah University is PRTG, which provides monitoring to the university branches and services. However, the used monitor system has a high financial cost, let alone it does not identify exactly the errors of the Application services.

This project focus on building an open-source monitoring tool that allows the university to monitor their services with a free budget. Furthermore, it gives the possibility to upgrade the proposed monitoring system with more advanced monitoring features.

We aim to use several protocols that are essential to manage and monitor on-premises and Cloud networks. Especially, the Simple Network Management Protocol (SNMP), Windows Management Instrumentation (WMI), and Internet Control Message Protocol (ICMP), which are considered the most known and used protocols in the network management field. Consequently, we decided to build our open-source monitoring system. This system is based on an active monitoring approach, which focuses on collecting the status of specific services in the university infrastructure. Finally, all the collected data will be illustrated and analyzed by a suited Graphical User Interface (GUI) dashboard.

KEYWORDS NMS; SNMP; WMI; Monitor; GUI.

ملخص

تعدّ أدواتُ مراقبةُ الشبكةِ أدواتٍ مهمةً للمؤسساتِ، لأنها توفرُ الاستقرارِ للشبكةِ والحمايةِ منْ توقفها. إنَ نظامَ إدارةِ الشبكةِ الذي تستخدمهُ جامعةُ طيبةَ هوَ نظامٌ (PRTG)، والذي يوفرُ المراقبةَ لفروعِ وخدماتِ الجامعةِ. ومعَ ذلكَ، فإنَ نظامَ المراقبةِ المستخدمِ تكلفتهُ الماليةُ عاليةً، ناهيكَ عنْ أنهُ لا يحددُ بالضبطِ أخطاءَ التطبيقاتِ بالشبكةِ.

يركزَ هذا المشروعِ على بناءِ أداةِ مراقبةٍ مفتوحةِ المصدرِ تسمحُ للجامعةِ بمراقبةِ خدماتها بشكلٍ مجانيٍ. علاوةٌ على ذلكَ، فإنهُ يوفرُ إمكانيةَ ترقيةِ نظامِ المراقبةِ بميزاتِ مراقبةٍ أكثرَ تقدما. نهدفُ إلى استخدامِ العديدِ منْ البروتوكولاتِ الضروريةِ لإدارةِ ومراقبةِ الشبكاتِ المحليةِ والشبكاتِ السحابية.

على وجهِ الخصوصِ، بروتوكولُ إدارةِ الشبكةِ البسيطِ (SNMP)، وبروتوكول لإدارةِ أجهزةٍ Windows الذي اسمهُ (WMI)، وبروتوكولَ رسائلِ التحكمِ في الإنترنت (ICMP)، والتي تعتبرُ البروتوكولاتُ الأكثرُ شهرةً واستخداما في مجالِ إدارةِ ومراقبةِ الشبكةِ. وبالتالي، قررنا بناءُ نظامِ مراقبةِ مفتوحٍ المصدرِ. يعتمدَ هذا النظامِ على طريقةِ المراقبةِ المستمرةِ، والتي تركزُ على جمعِ معلوماتِ الخدماتِ في البنيةِ التحتيةِ للجامعةِ.

أخيرًا، سيتمُ توضيحُ وتحليلُ جميعِ البياناتِ التي تمَ جمعها بواسطةِ لوحةِ معلوماتِ ملاءمةٍ لواجهةِ المستخدمِ الرسومية (GUI).

كلمات مفتاحيةٍ: بروتوكولِ إدارةِ الشبكةِ البسيطِ؛ بروتوكولٌ لإدارةِ أجهزةٍ Windows؛ بروتوكولِ رسائلِ التحكمِ في الإنترنت؛ واجهةِ المستخدمِ الرسومية؛ برمجياتٍ مفتوحةً المصدرِ.

ACKNOWLEDGEMENTS

First of all, we want to give all the gratitude to Allah (glory be to Him) for His blessing and guidance for us. We would like to express our deepest thanks to our supervisor Dr. Wael Alheadary for his guidance, caring, support, and providing us with excellent information for doing this project. Also, we would like to give our appreciation to the Deanship of Information Technology at Taibah University for their assistance and support, especially Eng. Maher Ezzi. Finally, we would like to thank Dr. Salah Abdelmageid and Dr. Orabi Shurrab for giving us useful comments and for their encouragement. They provided us with a lot of useful information about the practical aspect; To find out the services used in the management and monitoring field. A final thanks to our family for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY AND LIST OF ABBREVIATIONS

Abbreviation	Description
ADFS	Active Directory Federation Services
AGPL	Affero General Public License
AI	Artificial Intelligence
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DRY	Do not Repeat Yourself
FTP	File Transfer Protocol
GNU	GNU's Not Unix
GPL	General Public License
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IIS	Microsoft Internet Information Services
IMAP	Internet Mail Access Protocol
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
MIB	Management Information Base
NAS	Network Attached Storage
NMS	Network Management System
NOC	Network Operations Center
OID	Object Identifier
OOP	Object-Oriented Programming
OS	Operating System
OSI	Open Systems Interconnection
PaaS	Platform as a Service
POP3	Post Office Protocol
PRTG	Paessler Router Traffic Grapher
SaaS	Software as a Service
SHA	Secure Hash Algorithm
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
VM	Virtual Machine
Wi-Fi	Wireless Fidelity
WMI	Windows Management Instrumentation

CHAPTER ONE

INTRODUCTION

1.1 Overview

In the past few years, networks have become something significant for humanity, it rapidly and continuously grows every day with new technologies. Hence, network monitoring is essential to maintain the stability and confidentiality of network services. Particularly, in the Taibah University network, which has a lot of systems and servers, the demand for network monitoring appears especially during heavy traffic load. As a result, a lot of faults and problems will occur in the university services.

For the time being, Taibah University uses a PRTG monitoring system for network monitoring, but it lacks some features that facilitate problem identification. Accordingly, the final result of the project should help the university to decrease the financial cost, and to add more features and flexibility to detect Application errors. Additionally, it will guide us to add a depth knowledge about networks, programming, and several details of conducting academic scientific research.

1.2 Problem statement

Taibah University is using PRTG for monitoring their devices, it's expensive and it lacks some features; for example, the monitor does not send a detailed warning about which layer the problem is located in, especially from the Application perspective. Due to that, our project will fix this problem by developing a monitoring tool from scratch. The proposed tool monitors the layers by sending an alert that contains the fault and in which layer it can be exited. To be specific, we will focus on detecting service faults.

Network management systems in the network markets are divided into two main groups: commercial and open-source tools. By using a commercial tool, each feature of the monitoring tool needs an extra cost to be used as well as it has a limitation in its function to detect the Application faults. On the other hand, open-source tools are more flexible but it is unfriendly to be used and modified.

1.3 Motivation

After we have investigated all the offered projects which are presented by the computer engineering department, we found out that this project will help us to build a new

monitoring tool, which is a new experience for us. Besides, it will help us to learn more about:

1. Real networks infrastructure.
2. A new programming language.
3. Network managing and monitoring protocols.
4. Improve our search skills.
5. Give us the chance to use other operating systems like Windows Server and Ubuntu Server.
6. Working on complex projects.
7. Make an open-source software that can be extended to any new network technology and infrastructure.

Hence, we have chosen this project for these reasons. Honestly, we think it is a good opportunity and a great way to implement what we have learned in the past few years.

1.4 Project Goals

Our monitoring tool is able to detect the network faults of Taibah University, like websites crashing due to heavy load traffic or any other bugs.

Network monitoring protocols will help us to manage the network services. Let alone, it offers more effective solutions and extended to any new technology and any infrastructure.

1.5 Outline

This project is organized into several chapters. Chapter 2 provides background and related works, chapter 3 presents the project system design, chapter 4 covers the implementation, and chapter 5 is a conclusion, and future work.

CHAPTER TWO

BACKGROUND

2.1 Introduction

This chapter will discuss several topics in computer networks, such as the basics of networks, the Network layers, network protocols, socket programming. In addition to that, we will review some of the related works that we explored. Furthermore, we intend to present the methodology that we used to implement the proposed algorithm.

2.2 Computer Networks

A group of devices that communicate with each other, whether on a local or a wide area to share resources or Internet services. They have become a priority in every home, company, or university, due to their benefits that save time and effort for everyone. Hence, it has become an indispensable technology.

Computer networks are designed separately into seven layers. Therefore, each layer could be modified independently. In fact, these layers are useful in numerous things, like determining the required hardware and software to help in building networks.

In the next sub-sections, we will present the basics of network topology, which clarify the network elements distribution, and how the data are transmitted through layers and protocols between sender and receiver.

2.2.1 Basics in-network

Every computer network has basic components. Some of them are:

IP Addresses and Subnetting IP address: IP address is a numerical label that is assigned to each node in the network, and it is used by other nodes for communication and location. The IP address is a human-readable binary number. It includes the elements that make up the network and can be divided into different sub-networks based on device type, access, etc.

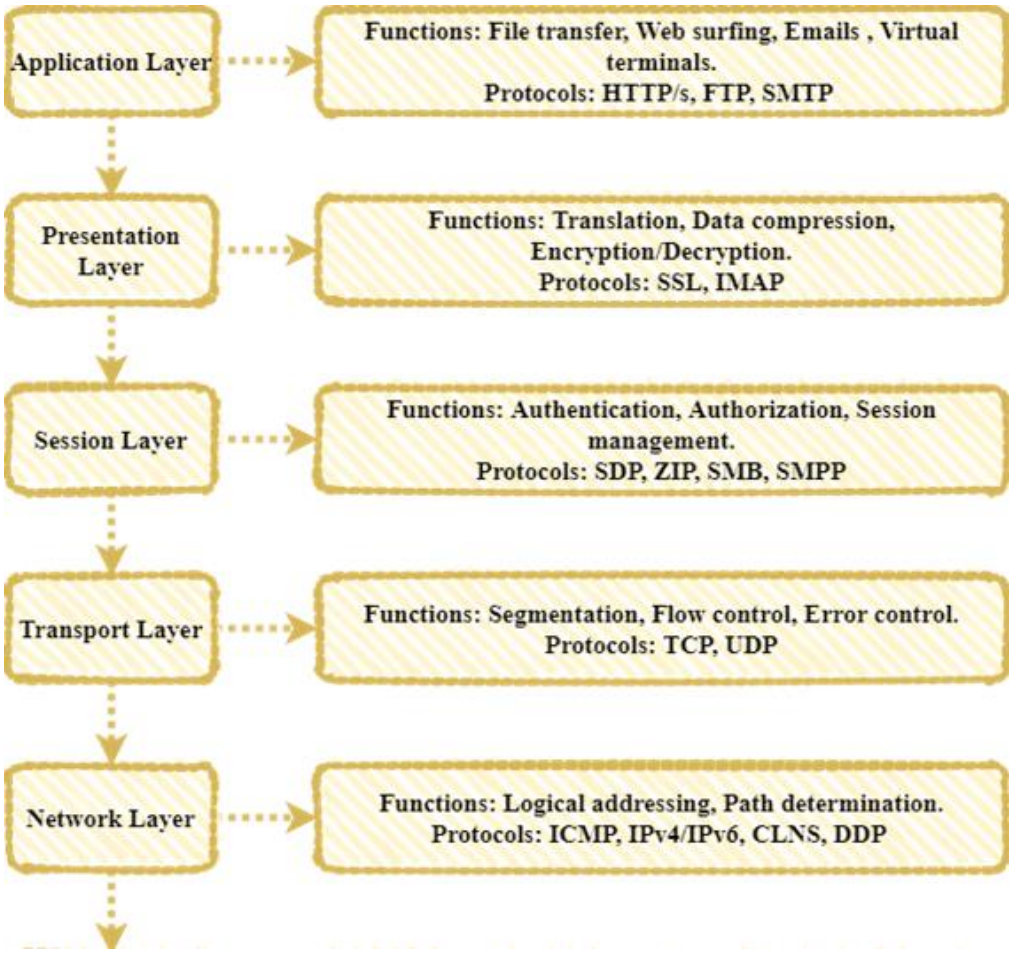
Domain Name System (DNS): Each element in a network has an IP address, but it is hard to remember every IP address, that is why we use reference names. This aids the user to use alphabetical names rather than numbers. DNS translates a physical IP address to a name, so anyone can use and remember it; it is also faster and understandable.

Dynamic Host Configuration Protocol (DHCP): It allows a DHCP server (a management server) to add an IP address dynamically to network resources. Furthermore, DHCP is a network protocol with a connectionless protocol it uses UDP [1]. Without it, network admins would have to add IP addresses manually; by using DHCP the IP address management is easier.

Switching and Routing: Switching is the process of data that is divided into small packets before it transports over the network. On the other, routing is an act of finding a packet path, and it is used to link multiple networks together.

2.2.2 Network layers

Networks are separated into seven layers based on the Open Systems Interconnection model (OSI model), each layer performs a particular function, such as troubleshooting network problems. Moreover, all these layers work together to transfer data between devices around the world. Notably, Figure 2.1 shows us the seven layers, their protocols, and their functions.



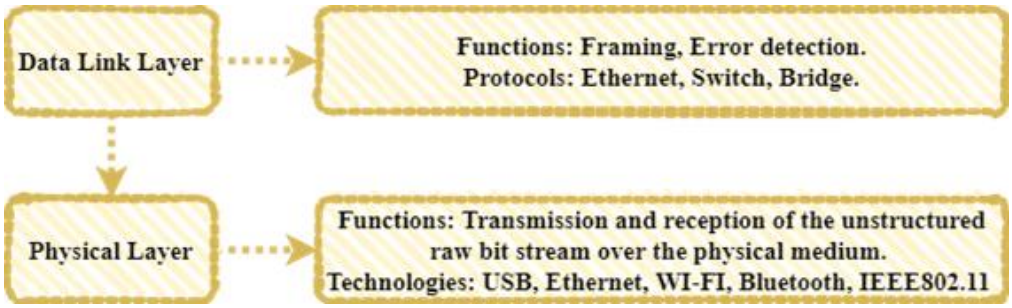


Figure 2.1: Network layers

2.2.3 Basic LAN Network topologies

Network topology refers to the various nodes and connections on a network, and if they are physical or logical.

Local Area Network (LAN) is a group of devices connected in a single physical location. In addition, topology is a physical and logical procedure of a network. Figure 2.2 shows a LAN topology, which explains how all the devices are connected to the internet [2].

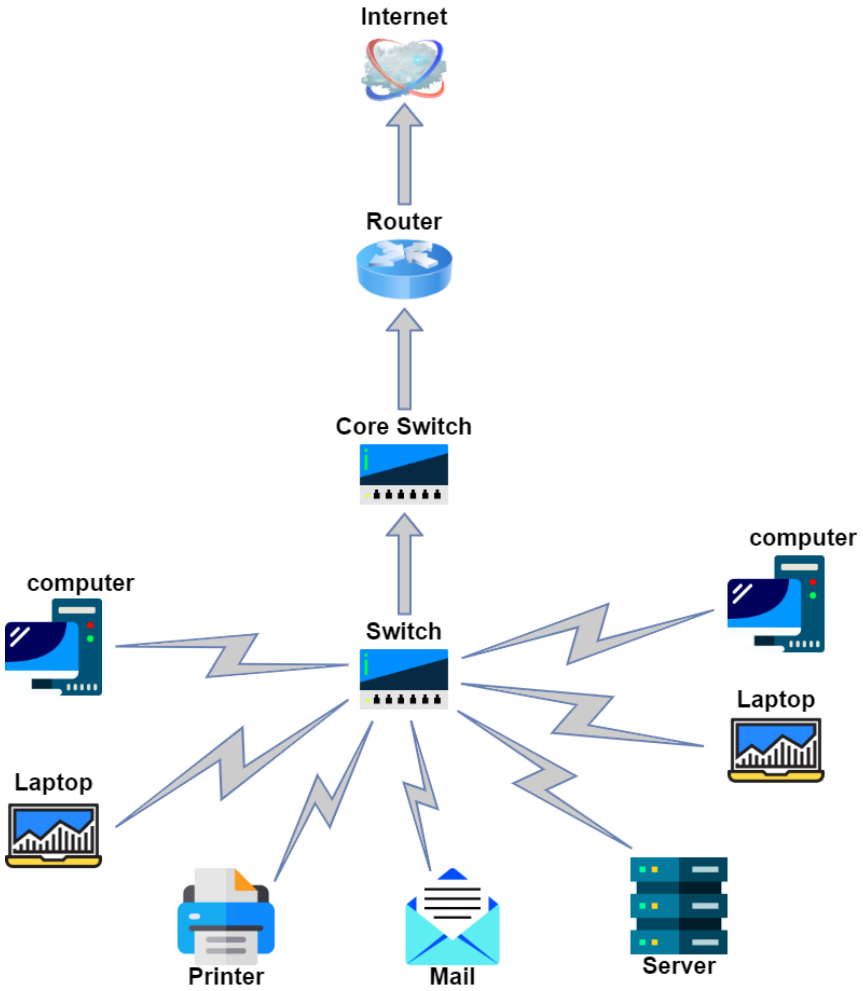


Figure 2.2: Basic Network Topology

2.2.4 Network Management System

Is an architecture for managing communications technology, networks, and services to detect a device failure in a network. Thus, by using network management tools, it continues to send reports to the network administrator and the NOC team. Let alone, a network administrator is the one that monitors services and analyzes the network traffics, and detects bugs [3].

Network management includes several services such as monitoring, configuration, analysis, and control of network element resources and elements for real-time operations [4].

NMS's main purpose is to detect network faults and send an alert about network problems.

2.2.4.1 Network Monitoring Approaches

Network monitoring approaches are divided into active and passive. Where Passive collects the user data and examines it in a specific period of time. On the other hand, Active monitoring determines possible network performance by simulating user behavior in a real-time network. Moreover, network monitoring infrastructures are split into an agent and agentless tools. Which depends on the nature of the monitoring system and the used protocol. Figure 2.3 shows a simple description of the monitoring system uses between various devices.

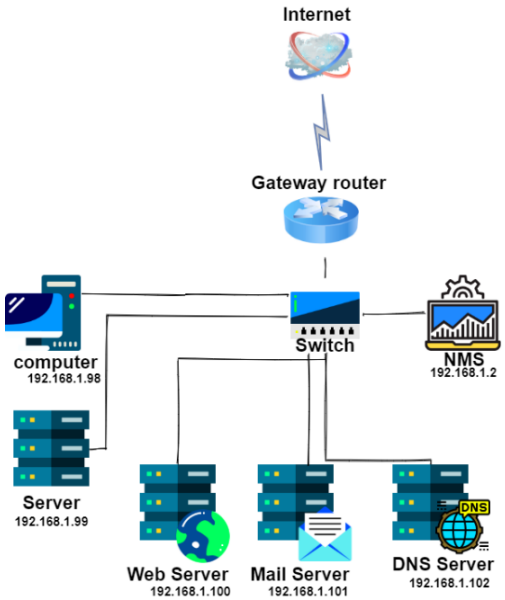


Figure 2.3: Monitored LAN

The monitoring process is categorized into three parts as follows [5]:

1. Monitoring of device availability: They care only about monitoring the availability of devices.
2. Monitoring of Service availability: They are based on monitoring the availability of Services on the Servers like Mail, Web-based functionalities, and other services operate in the Application Layer.
3. Monitoring of other parameters: These are the status of all ports of the switch, bandwidth, and checking the fan status to keep the device temperature stable; because not having to do CPU throttling.

2.2.5 Protocols

Protocols are a set of rules that directive how devices exchange data through networks. Moreover, each network's layer has its unique protocol that can do different tasks. We intend to review some of them in this section.

2.2.5.1 ICMP

The Internet Control Message Protocol is one of the internet protocols for network management and monitoring. Also, it is used to send error messages that will help to detect errors in the underlying communication of network applications [6].

Ping is one of the ICMP utilities, it is a lightweight utility because it has small packets that will give quick results. Add to that, it sends ICMP echo request packets and tests the availability of a device or host on a network like in Figure 2.4.

Indicators are necessary for network monitoring, such as latency measured by a ping response.

Figure 2.4: Echo request and replies to the devices and server

2.2.5.2 SNMP

The core of the SNMP is a simple set of operations that gives administrators the ability to change the state of some devices in the network [7], In the same way, a protocol can be used to manage different systems [8]. It allows the administrator to monitor network equipment state, and manually modify settings and configurations in the equipment.

SNMP contains two important concepts:

- Object Identifier (OID): Are anything and everything on a device can be monitored. To put it another way, SNMP has an OID. Which shows up as a group of numbers that has a meaning but it's hard to understand, and it looks like an IP address.

- Management Information Base (MIB): It is a text file that allows us to translate the OID numbers into understandable words.

SNMP Manager communicates with the managed device using C# libraries. Look at Figure 2.5.

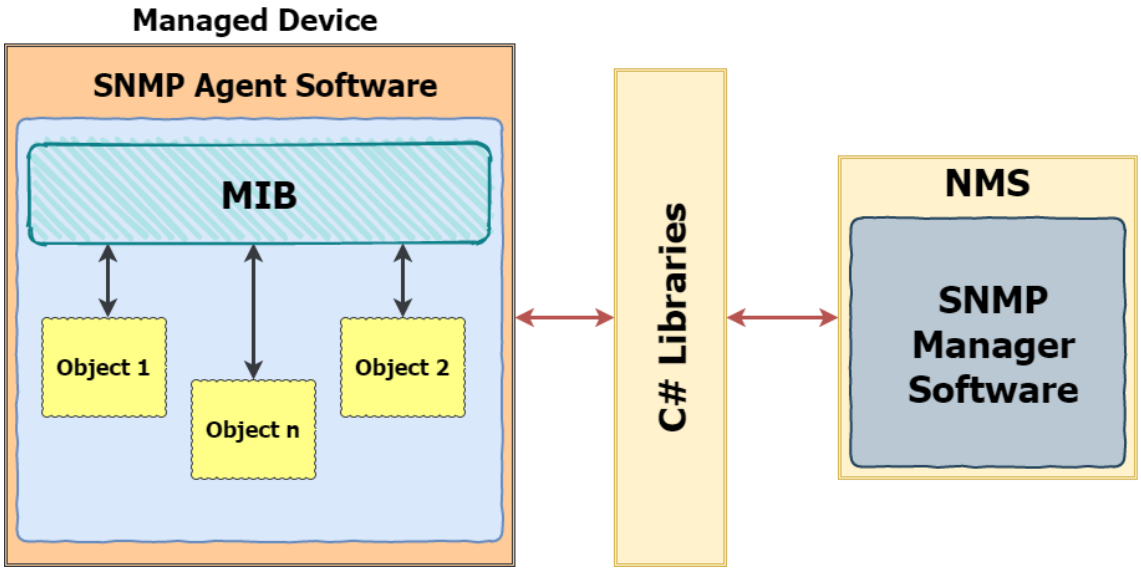


Figure 2.5: SNMP Manager communicates via C# Library

SNMP protocol contains two types of monitoring methods, which are Polling by query and trap.

The SNMP Polling is retrieving MIB variables from devices to determine fault behavior or connection problems. Moreover, It is connected to port 161 which tells the device which OID it wants the information on.

Trap allows network devices to notify the NMS about an event through an SNMP message. Moreover, it is connected to port 162.

As shown in Figure 2.6, SNMP Queries request from an NMS to network device and response from a network device to NMS as a reply to a request. On the other hand, the Trap is from a network device to NMS.

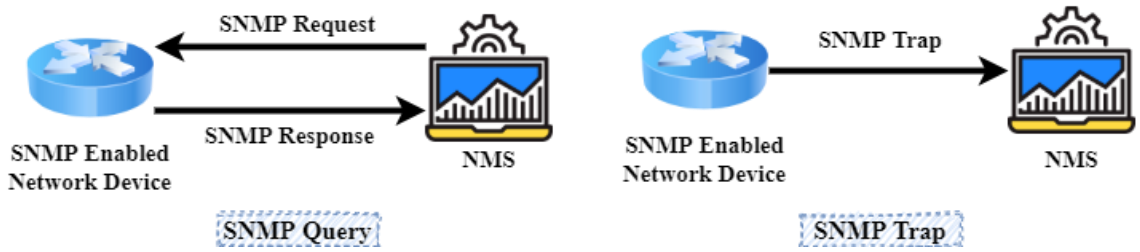


Figure 2.6: SNMP methodology

By using the SNMP protocol, different devices can be monitored from the network monitoring server, like what is shown in Figure 2.7.

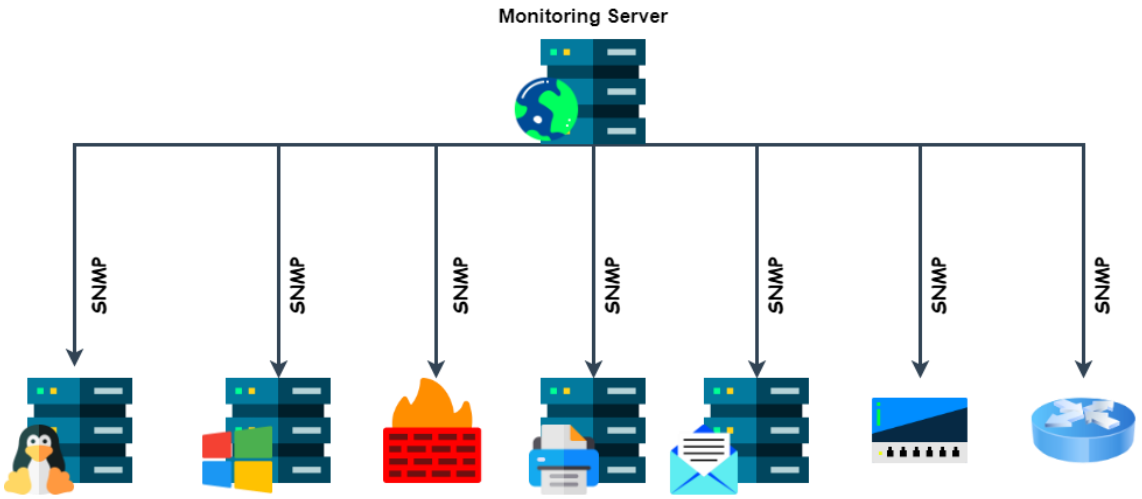


Figure 2.7: SNMP services

There are three versions of SNMP that are shown in Table 2.1. However, there are differences between them. Additionally, some devices do not support the latest versions, and this must be taken into consideration when using SNMP in NMS.

Table 2.1: SNMP versions

Versions	Level	Authentication	Encryption	Inform and Trap
V.1	noAuthNoPriv	Community string	No	Only trap is supported
V.2c	noAuthNoPriv	Community string	No	Supported
V.3	AuthPriv	MD5 or SHA	DES	Supported

2.2.5.3 WMI

The Windows Management Instrumentation is a set of specifications that are exclusively working on windows systems and has a set of extensions that provide an OS interface through NMS to show notifications and information [9]. Moreover, It enables programming languages such as C# to control computers and servers.

2.2.5.4 Comparison between SNMP and WMI

WMI is easy to set up, but it uses more resources to do the same tasks as SNMP. If the work involves many elements, SNMP is usually a better choice. Considering that [10] using SNMP to fetch information from the system reduces monitoring overhead, but can be difficult to configure.

2.2.6 Socket programming

Network programming uses sockets to communicate between layers. Actually, network programming is also known as Socket programming.

The processes are running on the devices with each other by sending and receiving messages as sockets [11]. Accurately choosing the programming language helps in programming networks, and in this project, we chose the C# language; because it supports several libraries and deals with the Internet by communicating and sending commands.

2.3 Comparison

Monitoring systems help companies and universities in analyzing their technologies and watch the operation and performance of them and assist in detecting and inform about errors.

This section is going to compare four of the monitoring systems out there. As shown in Table 2.2 a reference of some of the network monitoring and management services and a comparison between them.

Table 2.2: Monitoring tools comparison

Name	License	SNMP	MIB compiler	Platform
PRTG	Limited free, Commercial	Supported	Supported	Windows
Solarwinds	Commercial	Supported	No	.NET
Nagios	GNU GPL	Via plugin	No	C
OpenNMS	AGPLv3	Supported	Supported	Java
Zabbix	GNU GPL	Supported	No	C, PHP
Opsview	Commercial	Supported	No	Perl, C, ExtJS
Icinga	GNU GPL	Via Plugin	No	C

2.4 Related work

This section reviews some related and relevant research to the domain of network management and monitoring systems.

2.4.1 Overview of monitoring

The basics of networks monitoring that were referenced in [5] show that any tool or skill for network monitoring will provide a footstep guide on how to develop a network programmer as well as a host device network monitoring tool, with a mention of the main elements that we focus on it in monitoring and methods across Java libraries.

2.4.2 Network Monitor Approaches

Network Monitor Approaches as referenced in [12] offers an overview of current approaches that can be either active or passive to network monitoring, their architectures, functions, and properties. It also compares these approaches. As well, it must be

configured effectively with what mentioning in references [13], which explained what provides an effective and automated network management and monitoring system that immediately informs network administrators in the event of a problem. Additionally, the presented network monitoring system is quick in identifying the location of problems in the network and their impact on other parts of the network. Therefore, it is highly efficient and gives you full control over your network.

2.4.3 Use of open-source monitoring systems

Some developers in network management centers prefer to use open-source tools. Reference [14] describes a helpful system that monitoring the network topology continuously, and comparing it with Zabbix and OpenNMS.

The project consists of software that has a good history of dynamic development to the flexibility of software design. Furthermore, some of the main results included server metrics like CPU status, memory usage, and different time indicators in performance graphs obtained with the support of the plug-in.

Reference [15] shows that the network is monitored based on various parameters. Therefore, in different organizations. Also, the software tool reduces effort by monitoring all servers from the same main server. Furthermore, various tools are available to monitor a specific part of the servers and are modified according to requirements. In this system, we have seen briefly the software tool to be designed like Open Source, which has been created in the past, with features that are easy to understand and come with great performance, and about the scope and limitations of these systems such as Nagios, Hyper HQ, Open NMS, Zabbix, Zenoss, etc.

2.4.4 Building the program from scratch

Some programmers prefer to create their system from scratch, regarding [16] , which shows that there are three categories: sensors, probes, and services. Were Sensors are the lowest level monitoring objects, and probes are persistent objects, which maintain the information, and services are the highest level monitoring objects. besides, the web interface makes it easy to re-establish the embassy, which will solve critical issues found in existing applications.

As a part of the methodology, the project consists of three parts, which are web and database servers, Network Attached Storage (NAS). We will monitor local parameters: Disk, Memory, and Standard TCP services and state of applications.

2.5 Methodology

In this section, our methodology includes the requirements analysis of a project sequence to achieve the goal of detecting network application problems. This plan is significant for developing the project and the program in their final form.

2.5.1 Analysis Models

The development of software projects is one of the most important, and powerful issues for computers. Still, the system software development process is a very complex thing without a proper step-by-step production process. Honestly, this software is a systematic and structured method of the software development process.

In Figure 2.8 we used the Prototype model, which allows us to first analyze the requirements, and this is what we did in the previous sections, then we find what is required in writing the algorithm and services for the design stage. being that, the reason that we chose the Prototype model is to continue evolving the project, updating it, and designing it in a more organized way.

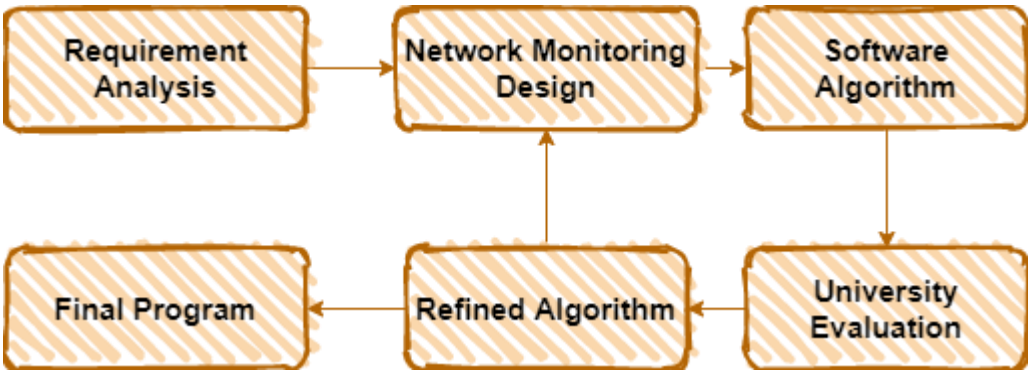


Figure 2.8: The Prototype Model

2.6 Project Timeline

The following Tables 2.3 and Table 2.4 show the progress of the project during the previous weeks. So that it shows the things that have been focused on every week, whether from tasks, planning, or documentation.

Table 2.3: Project 1 timeline

	Activity by Week from the start of the project															
Semester	First Semester															
Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Project Planning																
Requirement Engineering																
Literature Review																
System Analysis & Design																
System Construction																
System Testing																
Documentation																

Table 2.4: Project 2 timeline

	Activity by Week from the start of the project															
Semester	Second Semester															
Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Project Planning																
Requirement Engineering																
Literature Review																
System Analysis & Design																
System Testing																
Documentation																
Test in real infrastructure																

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 Introduction

This chapter discusses our system design. Therefore, the project aims to design and implement a suitable algorithm that monitors on-premises and Cloud networks.

Primary, this chapter will show a description of the component's services of the subsystems. Second, will display a complete algorithm in a flow chart, and will show the GUI proposed design. Finally, the network monitoring system will monitor different services and showing data in graphs and charts, and alarm systems.

3.2 Requirements

The requirements of the network monitoring tool can be summarized with the following points:

- Can detect faults at any layer of the network.
- Easy-to-use GUI.
- Has the ability to update and improve over time.
- The notifications feature for reporting and sending the report to our system email or SMS messages.

3.3 Proposed services for monitoring

The services that we focused on and utilized in the monitoring tool that we created were implemented in the graduation project 2. Thus, we intend to review the services in this section. Moreover, after we searched, we found the differences in the use of the services between different systems. Look at Figure 3.1.

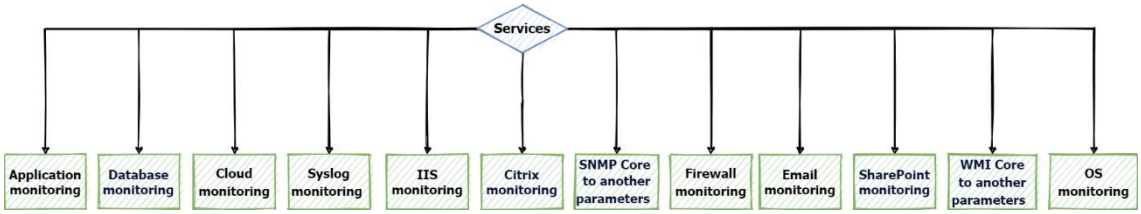


Figure 3.1: The services

3.3.1 Services based on SNMP

SNMP works with multiple devices from different platforms and manufacturers.

3.3.1.1 SNMP Core to other parameters

SNMP monitoring is valuable for servers and network devices, such as hosts, routers, and switches [17]. Therefore, this allows network usage, bandwidth, and problems such as downtime and traffic loads to be tracked and sending commands to manage the network.

3.3.1.2 Firewall monitoring

The stability of the network Firewall is responsible for protecting the ongoing processes and sensitive data in the university system, like grades and student information. Thus, Firewall Monitoring works with most routers and Firewalls and different companies such as Cisco and systems like Windows Server.

3.3.1.3 Email monitoring

Email monitoring is an important piece of a university. Also, it has other services, such as server availability, protocol availability that contain Internet Message Access Protocol (IMAP), and Post Office Protocol (POP3).

3.3.1.4 IIS monitoring

Internet Information Services (IIS) monitors the windows web server in the Application layer, and it detects faults. Then it warns the administrator about the problems so he can take the right action in time.

3.3.1.5 Citrix monitoring

Citrix systems use Virtual Machines (VM) to provide an optimized performance [18].

Taibah University uses virtual labs through Citrix Systems with the hypervisor, which provides paid software for teachers and students to use. Moreover, it runs applications through Cloud services.

3.3.2 Services based on WMI

A group of devices and technologies that generally rely on Microsoft Windows. For example, our university, which uses these services for the student's website and university mail in the University domain. These devices are using WMI.

3.3.2.1 WMI Core to other parameters

WMI monitoring gives detailed information about the servers and workstations that are running with Windows. By using WMI, it possible to access numerous Windows system state settings and configurations. Furthermore, it can be used to monitor services like Active Directory, SharePoint, or Database. And it provides indicators metrics such as CPU utilization, Memory Load, or Disk usage, Bandwidth Usage.

3.3.2.2 SharePoint monitoring

SharePoint is one of Microsoft's services for creating websites. Which Taibah University uses to develop their websites. It is possible to monitor a SharePoint server by using the WMI protocol. Thus, the number of current page requests is the indicator that affects active threads, Structured Query Language (SQL) queries, active cache size, global heap size, and CPU utilization, in real-time to stay in available service [19].

3.3.3 Public services

There are some necessary services to be present in the terms of network stability and security, and they can be used without a specific protocol by using socket programming libraries.

3.3.3.1 Application monitoring

Application monitoring ensures that all Applications (software or web) are performing as expected. This method evaluates the performance of applications and checks if everything is working accurately.

There are two different types of Application Management which are: standard Application management, and individual web Application management. Hence, Application monitor can monitor Standard Software such as databases, mail servers, Firewalls, and File Transfer Protocol (FTP).

3.3.3.2 Database monitoring

Databases are the main operator of the website. Furthermore, database monitoring has main factors including availability, downtime, traffic rates, and read and write rates.

Furthermore, the university depends on databases from Oracle, Microsoft, or IBM. Besides, various libraries are available to deal with databases.

3.3.3.3 Syslog monitoring

Network devices continue to record system messages through Syslog, these messages must be analyzed by administrators and the NOC team. In addition, proper monitoring tools are needed to monitor the Syslog. Therefore, you can quickly fix the bugs and solve security weaknesses [20]. By the way, Syslog is called Eventlog in Microsoft Windows systems.

3.3.3.4 Cloud monitoring

The world today is moving more than ever to Cloud services for storage and processing. At the time being. The most prominent software that can use Cloud service is SaaS, IaaS, and PaaS [21]. Likewise, it has support from multiple companies such as Microsoft and Amazon. In our university using Microsoft Azure Cloud.

3.3.3.5 DNS monitoring

The role of DNS monitoring is to inspect the server and obtain the IP address of any website we wish to be monitored.

3.3.3.6 OS monitoring

The role of the Operating System (OS) monitoring is to make sure that the resources in the device do not reach high numbers to weaken or freeze the server, as it monitors the processor, memory, storage, and details related to them.

3.4 General Algorithm

Algorithms should exist to know the project's workflow and what it requires from features and functions. Thus, we create an algorithm so that it can help us developing the monitoring tool. Furthermore, the algorithm is intended to be implemented by using socket programming, look at Figure 3.2. The benefit of making the algorithm in flow charts is to show the individual stages of the process sequentially and to describe different services and logics to the program.

We used the services that we mentioned in the past subsections in the algorithm, and we will discuss them in the next sections.

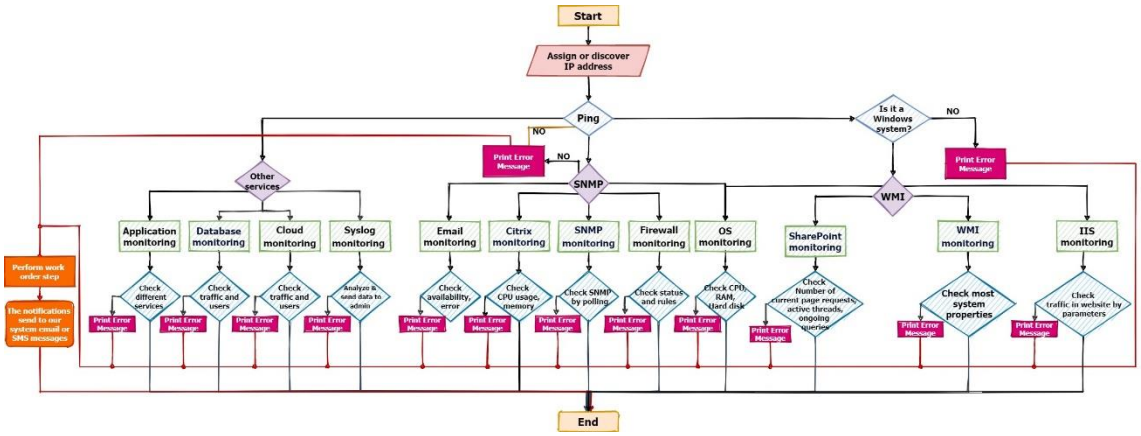


Figure 3.2: Our general flowchart algorithm

3.5 Early GUI Design

Graphical User Interface (GUI) is a computer software interactive visual component. In the early testing, we decided to create a GUI. To do that, we used Visual Studio 2019 software with C# language and its special libraries for networks. Then, we started programming without an interface through the Console App. After that, we create a GUI as shown in Figure 3.3. Besides recently we have started preparing to create a complete dashboard with interfaces as in Figure 3.4.

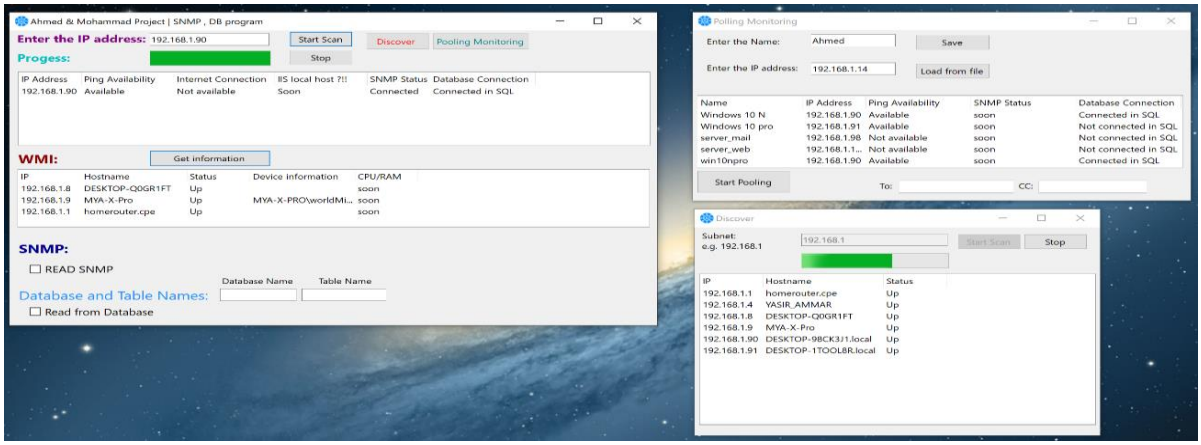


Figure 3.3: Old GUI of our program

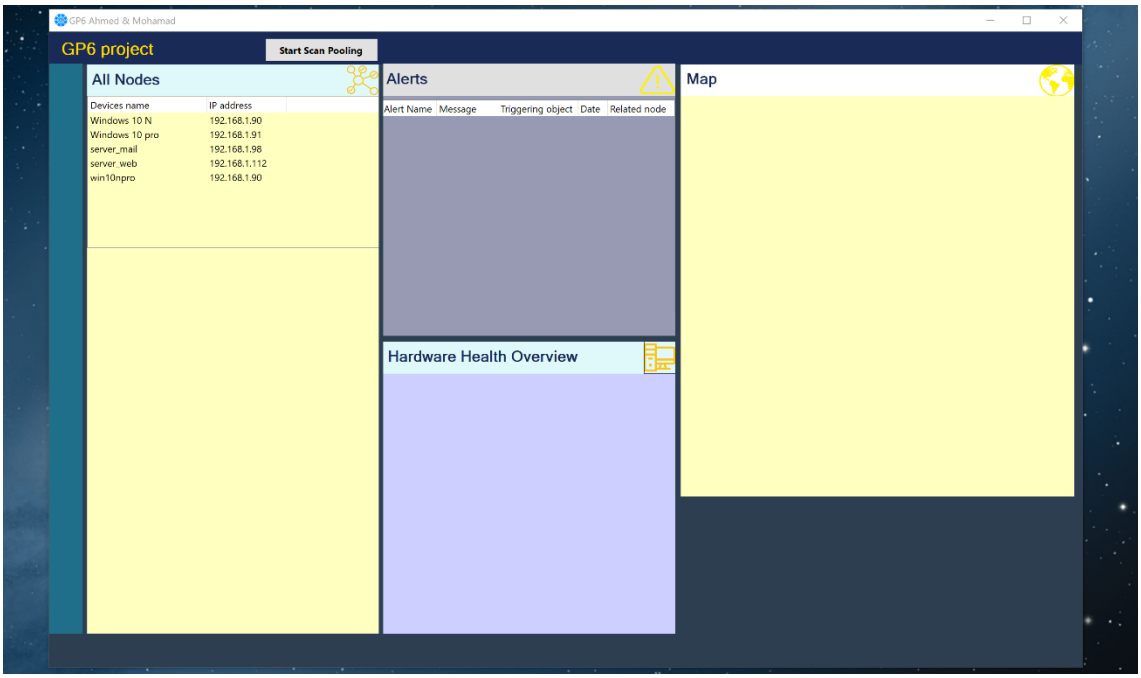


Figure 3.4: Dashboard beta version

3.6 System Analysis

3.6.1 Use Case Diagrams

To start using the monitoring tool, the admin user should log in to the program, after that the GUI will pop up and he will be able to view the services and start monitoring the network, for more information look at Figure 3.5.

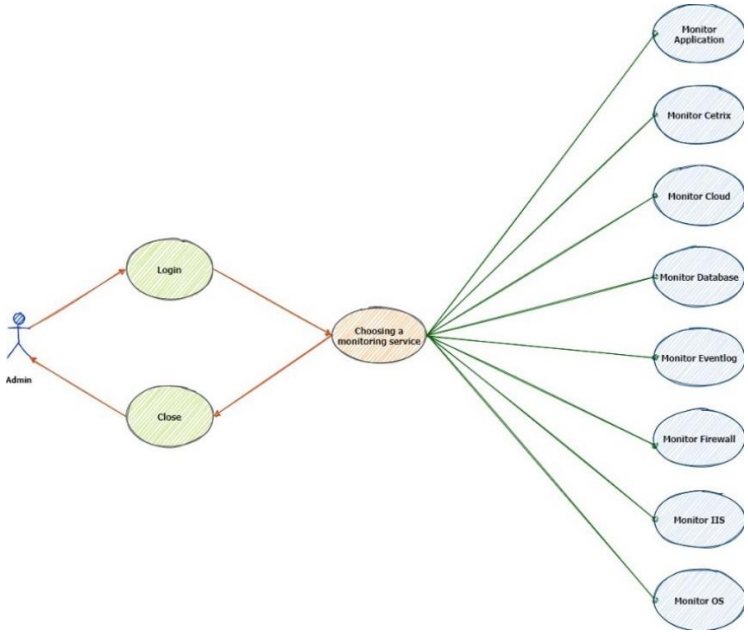


Figure 3.5: Our software tool monitoring use case

3.6.1.1 OS monitoring

The admin user, after logging in, can monitor OS by choosing the OS button on the user interface screen. Thus, by picking this service he can monitor the CPU, Hard drives, Network, and RAM. Look at Figure 3.6.

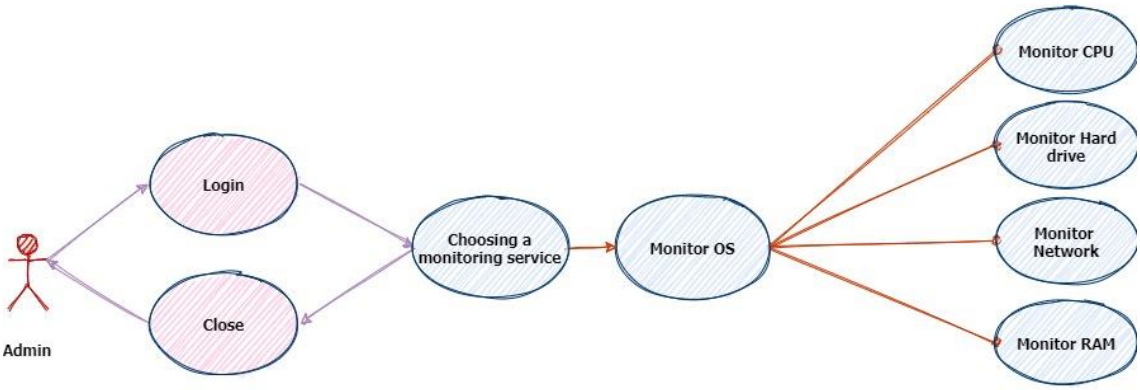


Figure 3.6: OS monitoring use case

3.6.1.2 IIS monitoring

The admin user, after logging in, can monitor the IIS by selecting the IIS button on the user interface screen. Thus, by choosing this service he can monitor the request traffic, files traffic, CGI traffic, and users traffic. Look at Figure 3.7.

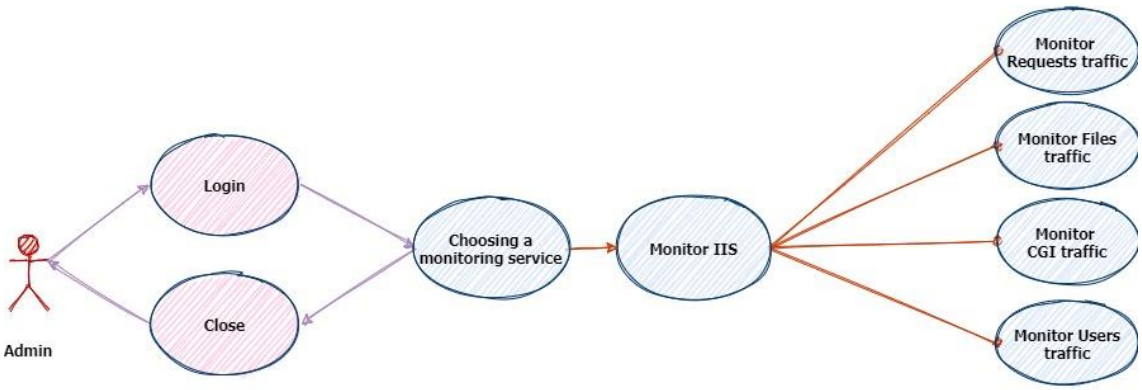


Figure 3.7: IIS monitoring use case

3.6.1.3 Firewall monitoring

The admin user, after logging in, can monitor the Firewall by choosing the Firewall button on the user interface screen. Therefore, by selecting this service he can monitor the TCP port scanning technique and (MSFT NetFirewallRule). Look at Figure 3.8.

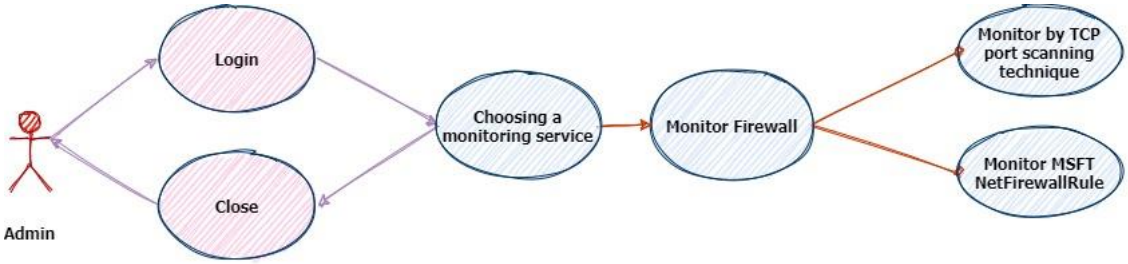


Figure 3.8: Firewall monitoring use case

3.6.1.4 Database monitoring

The admin user, after logging in, can monitor the Database by selecting the Database button on the user interface screen. Therefore, by selecting this service he can monitor the average time for reading and write, average time for insert and fetch, average waiting time, number of count star and fetch, and user information with current and total connections. Look at Figure 3.9.

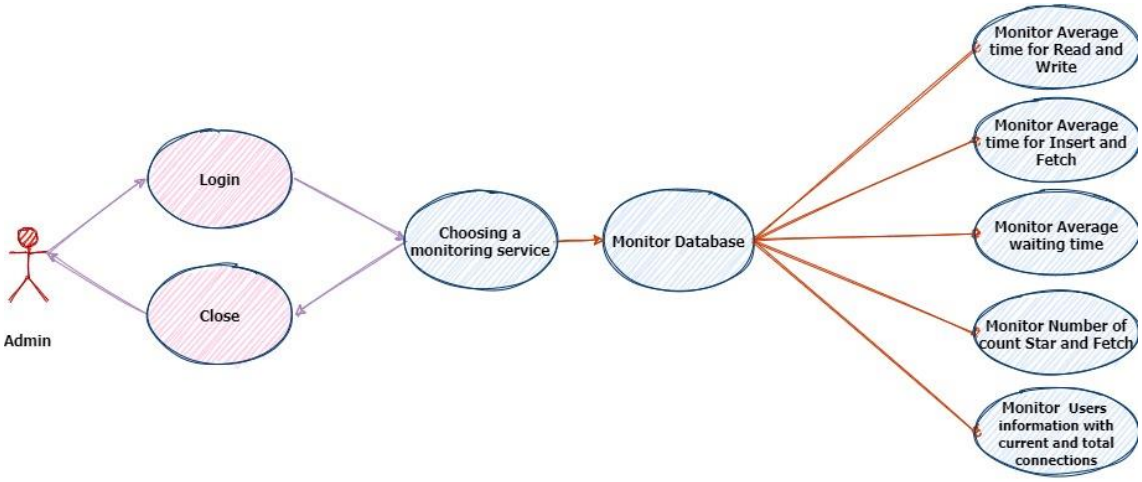


Figure 3.9: Database monitoring use case

3.6.1.5 Application monitoring

The admin user, after logging in, can monitor the Application by choosing the Application button on the user interface screen. Thus, by selecting this service he can monitor SMTP and FTP. Look at Figure 3.10.

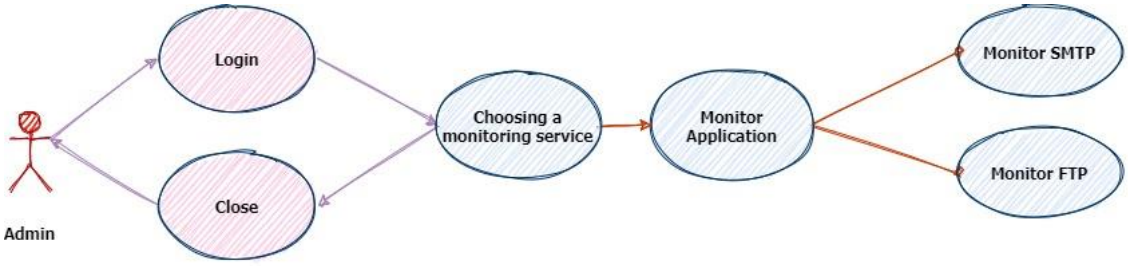


Figure 3.10: Application monitoring use case

3.6.1.6 Eventlog monitoring

The admin user, after logging in, can monitor the Eventlog by choosing the Eventlog button on the user interface screen. Hence, by selecting this service he can monitor the Messages per origin and severity level, and time and computer name. Look at Figure 3.11.

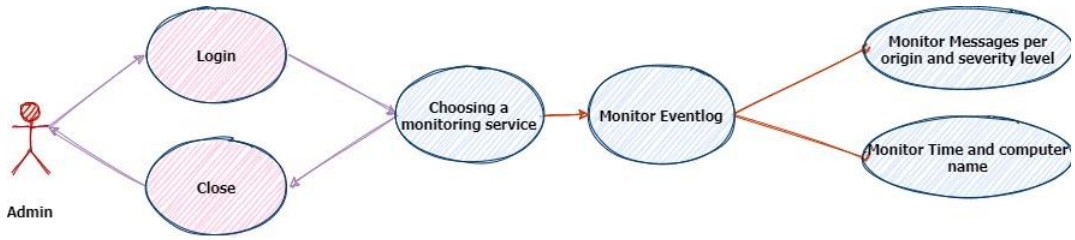


Figure 3.11: Eventlog monitoring use case

3.6.1.7 Cloud monitoring

The admin user, after logging in, can monitor the Cloud by selecting the Cloud Azure button on the user interface screen. Therefore, by choosing this service he can monitor the average time for reading and write, average time for insert and fetch, average waiting time, number of count star and fetch, and user information with current and total connections. Look at Figure 3.12.

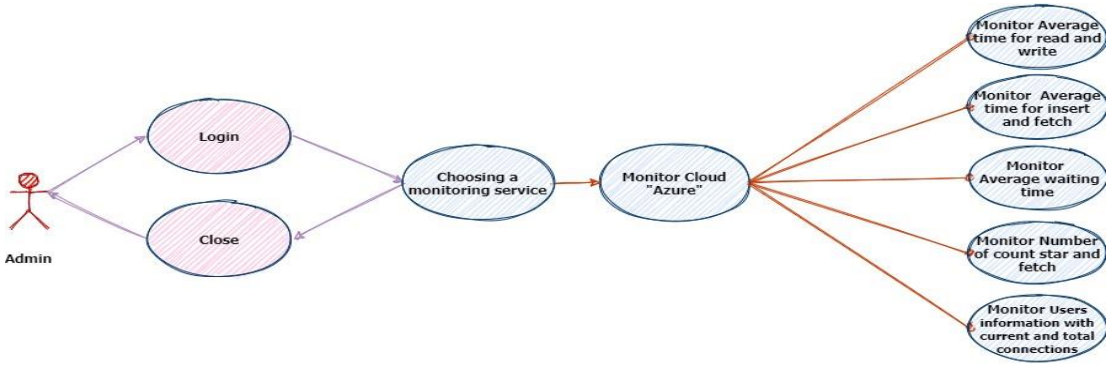


Figure 3.12: Cloud monitoring use case

3.6.1.8 SharePoint monitoring

The admin user, after logging in, can monitor the SharePoint by choosing the SharePoint button on the user interface screen. Thus, by selecting this service he can monitor the health of the server and the rate of the resources like CPU. As for the basic function of monitoring the number of requests and associated web applications. Look at Figure 3.13.

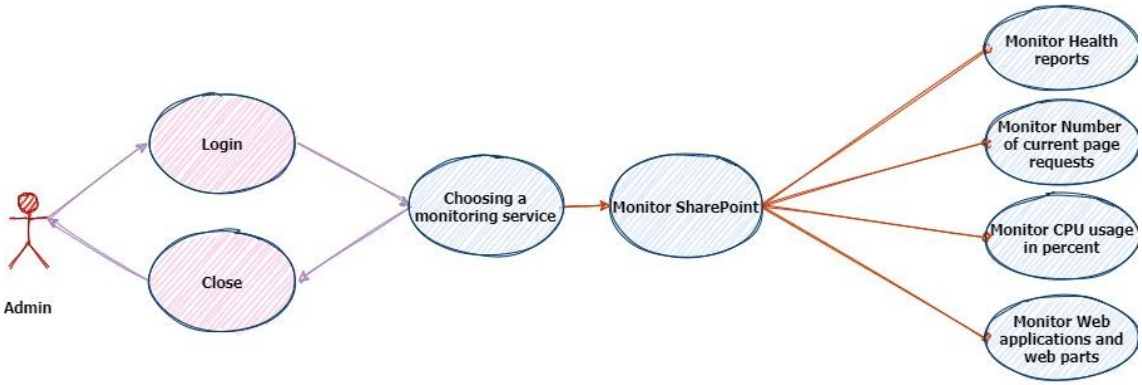


Figure 3.13: SharePoint use case

3.6.1.9 Citrix monitoring

The admin user, after logging in, can monitor the Citrix by choosing the Citrix button on the user interface screen. Thus, by selecting this service he can monitor OS usages like CPU and memory, and it is effectively available through active sessions to distinguish what programs use most of the server farm and power supplies. Look at Figure 3.14.

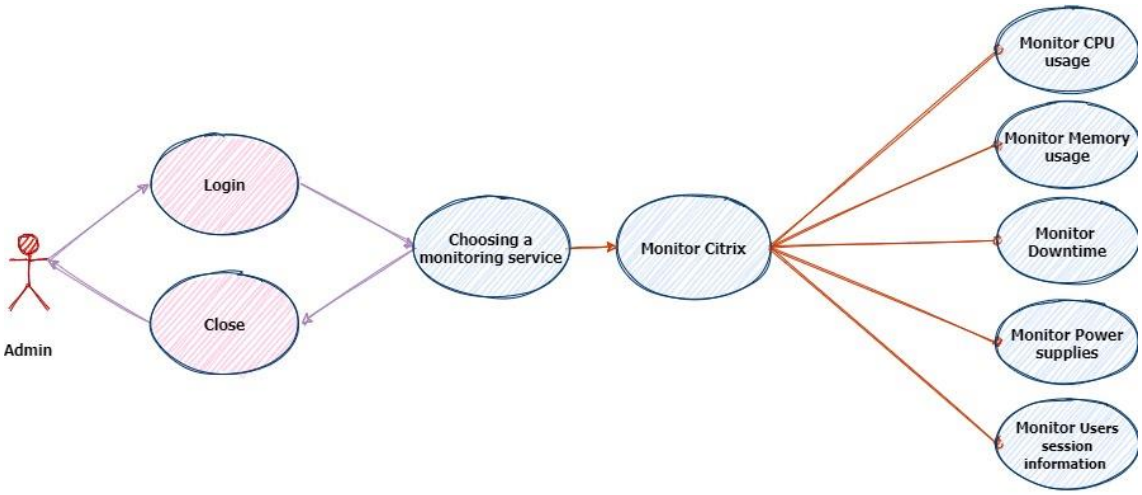


Figure 3.14: Citrix use case

3.6.1.10 Notification

To monitoring a system, the admin user of the system must log in to be able to receive notification messages warnings, if one of these cases occurs (server unavailable, access is denied, another problem from handling technology, server-status) it will send a notification. Look shown Figure 3.15.

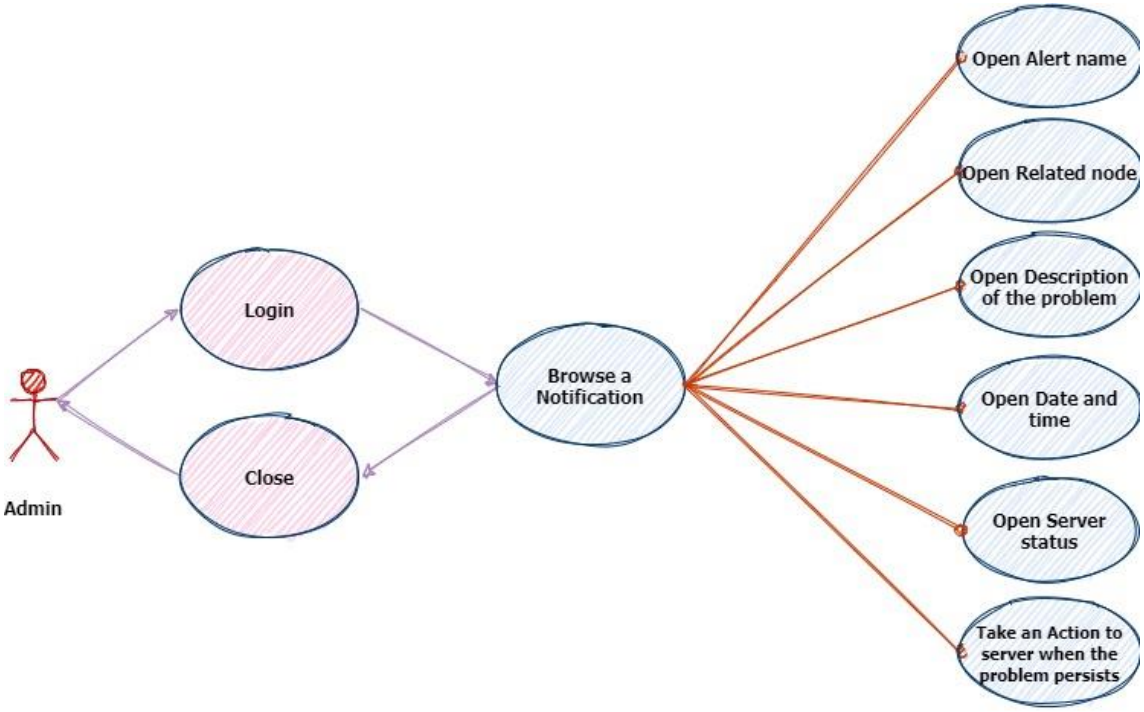


Figure 3.15: Notification Use Case

3.6.1.11 System Use Case

To monitor and management the system, the admin user of the system must log in to be able to view the current and previous status of the whole system as shown in Figure 3.16. More details are given above.

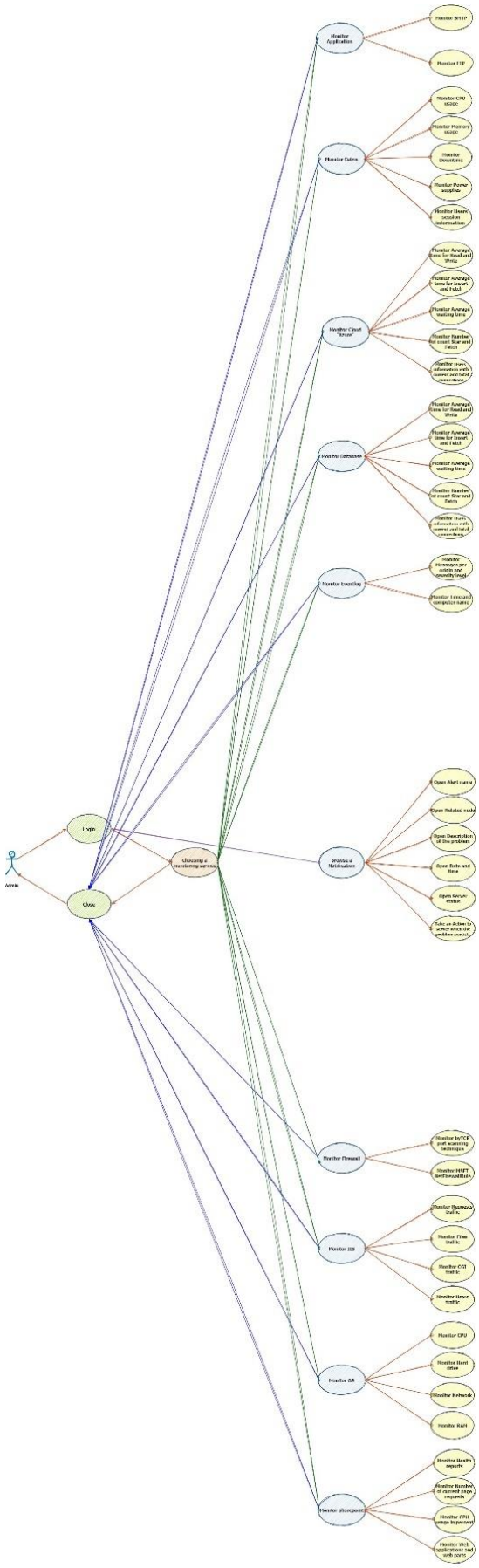


Figure 3.16: System Use Case

3.6.2 Activity Diagrams

WMI plays an essential role in our implementation, which deals with the Microsoft Windows platform. On the other hand, some operations use SNMP, and sometimes we resort to multiple techniques such as the TCP port scanning technique.

3.6.2.1 OS monitoring diagram

This flowchart explains the monitoring process of OS. Therefore, it gets the data like CPU, RAM, Hard disk, and Network. In the beginning, we should connect to a server or a device by using a username and password, and IP address which is mandatory for WMI services. Then, search for required parameters in query collection. Finally, check if the device has a high utilization to send an alert to the administrator. In either case, the program depends on being one-time or continuous monitoring. Figure 3.17 shows all details.

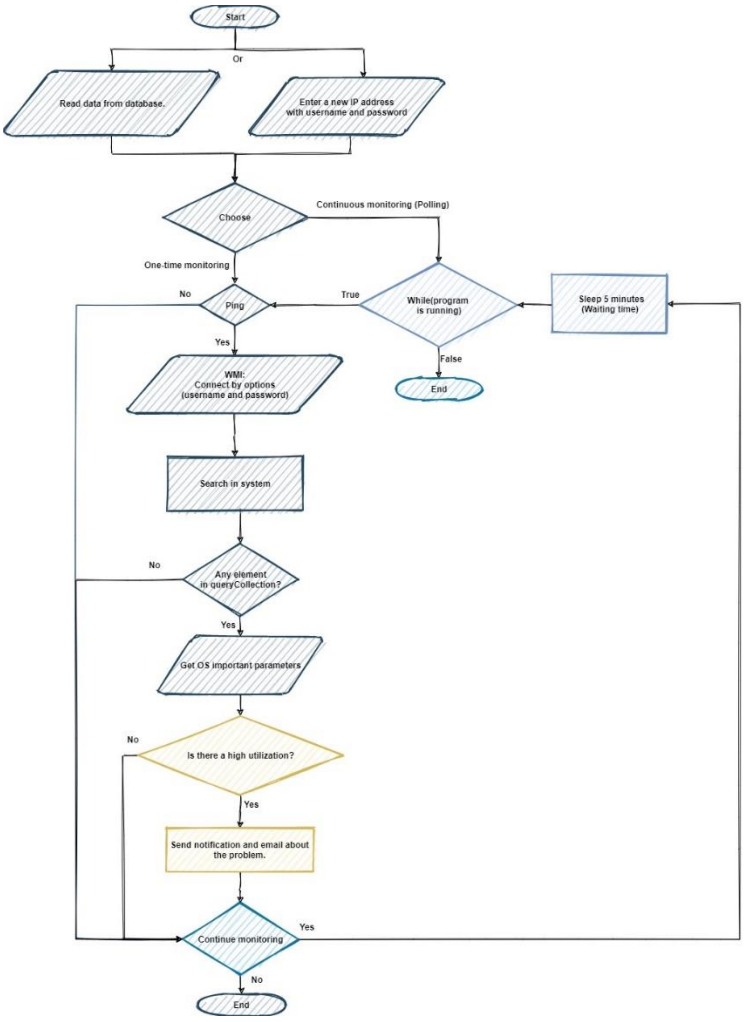


Figure 3.17: OS monitoring flowchart

3.6.2.2 IIS and DNS monitoring diagram

This flowchart explains the monitoring process of the IIS and DNS and HTTP. Moreover, it gets the data from the webserver hosted by IIS and imported the parameters like the Total Get Requests, Head, Files, and Error. This will implement for DNS as well, but it will get the host address of the website. And report by the HTTP request. Then, search for parameters in query collection. Finally, check if the server has high traffic to send an alert to the administrator. Figure 3.18 shows all the details.

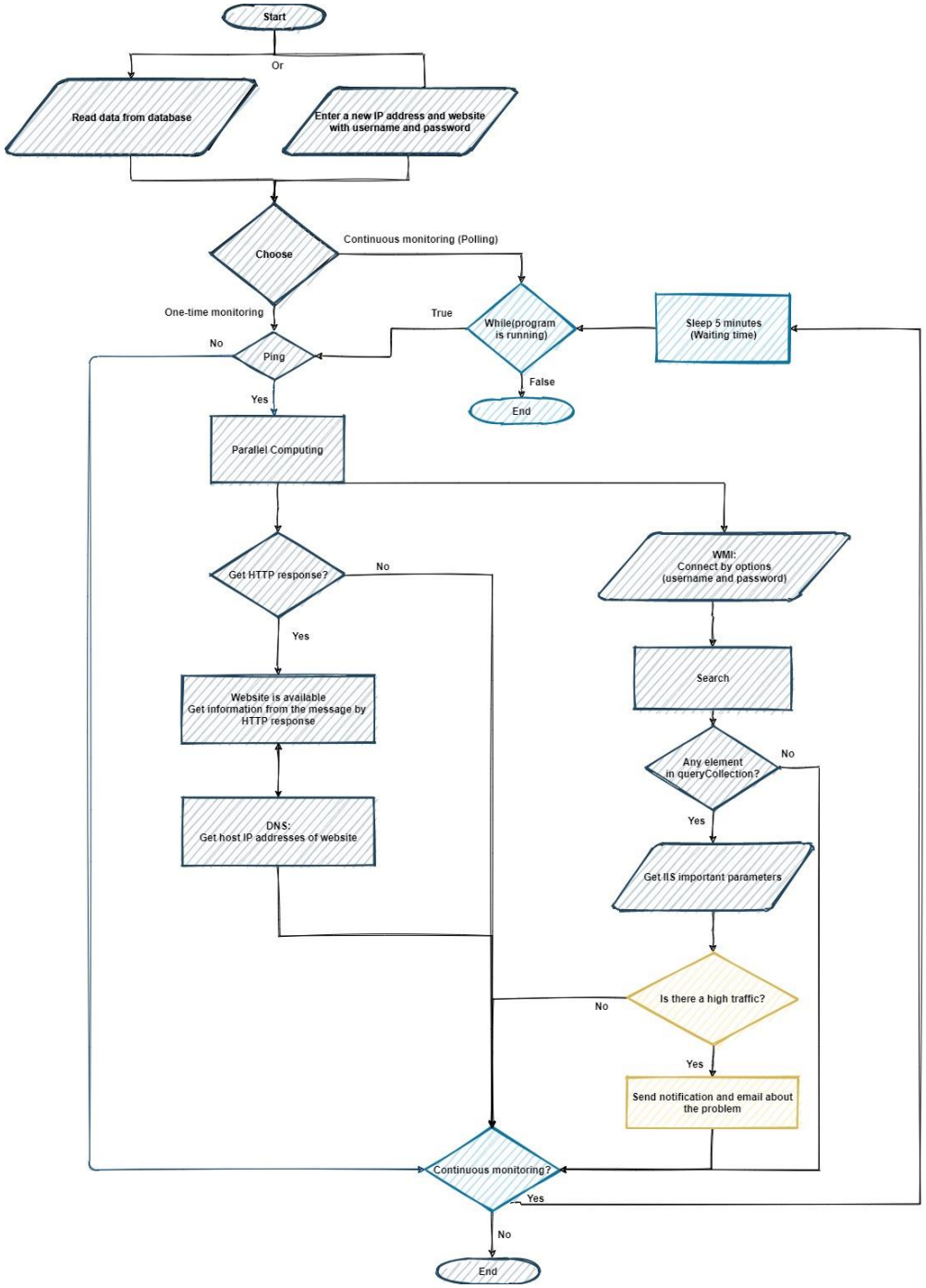


Figure 3.18: IIS and DNS monitoring flowchart

3.6.2.3 Firewall monitoring diagram

This flowchart explains the monitoring process of the Firewall. Therefore, it gets the data from the Firewall rules and policy of events recorded in the server. At first, we should connect to the device by using a username and password, and IP address which is mandatory for WMI services. On the other hand, make use of the TCP port scanning technique to check if important ports are available or have a problem. The completion of the program depends on it being one-time or continuous monitoring. Figure 3.19 shows all details.

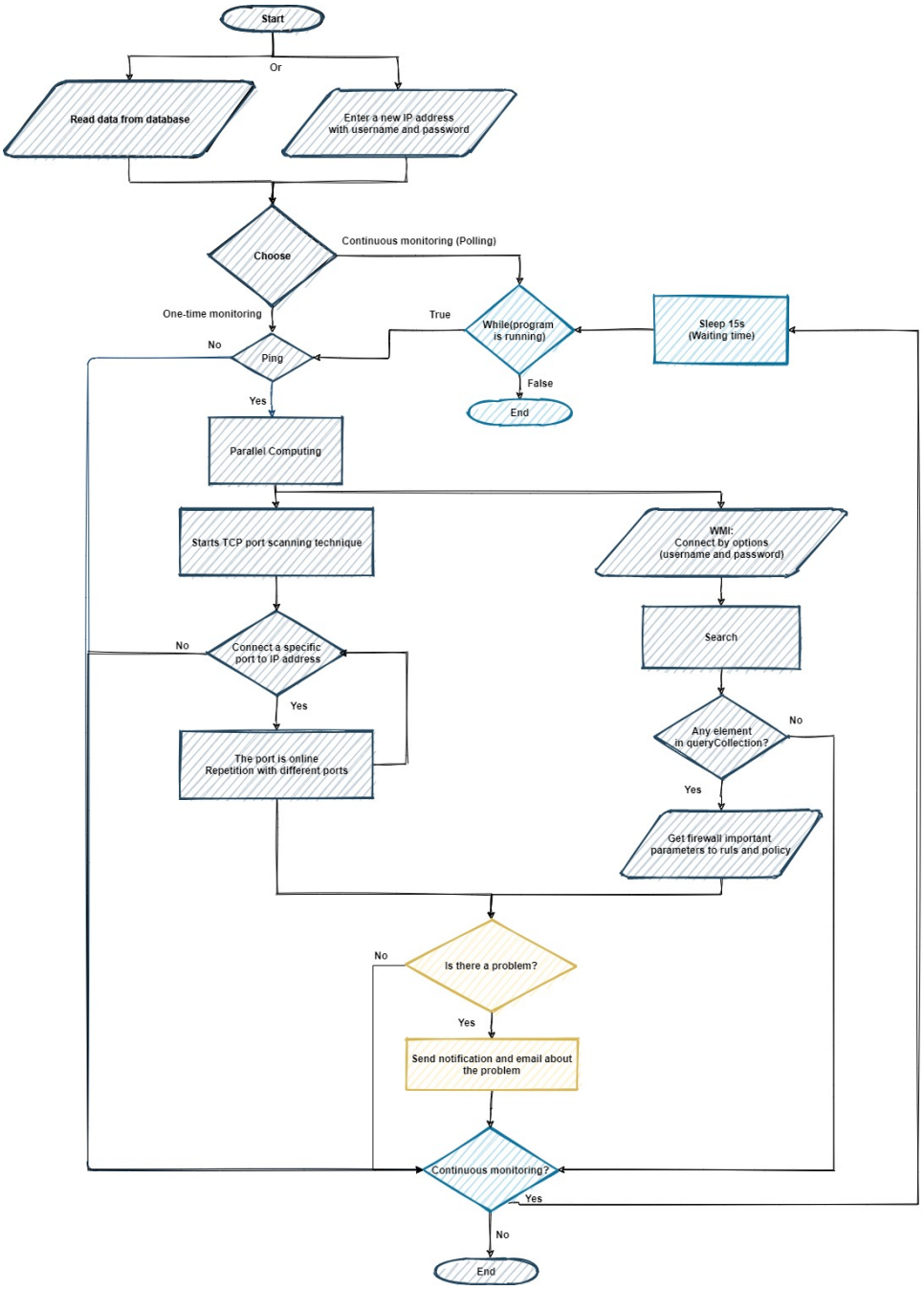


Figure 3.19: Firewall monitoring flowchart

3.6.2.4 Database monitoring diagram

This flowchart explains the monitoring process of the database. Therefore, it gets the important parameter in the database which is (performance-schema). At first, we should connect to the server by using a username, password, and IP address. We used a specific query to get the average for reading and writing operations, as well as the user information with the current and total connection. The completion of the program depends on it being one-time or continuous monitoring. Figure 3.20 shows all details.

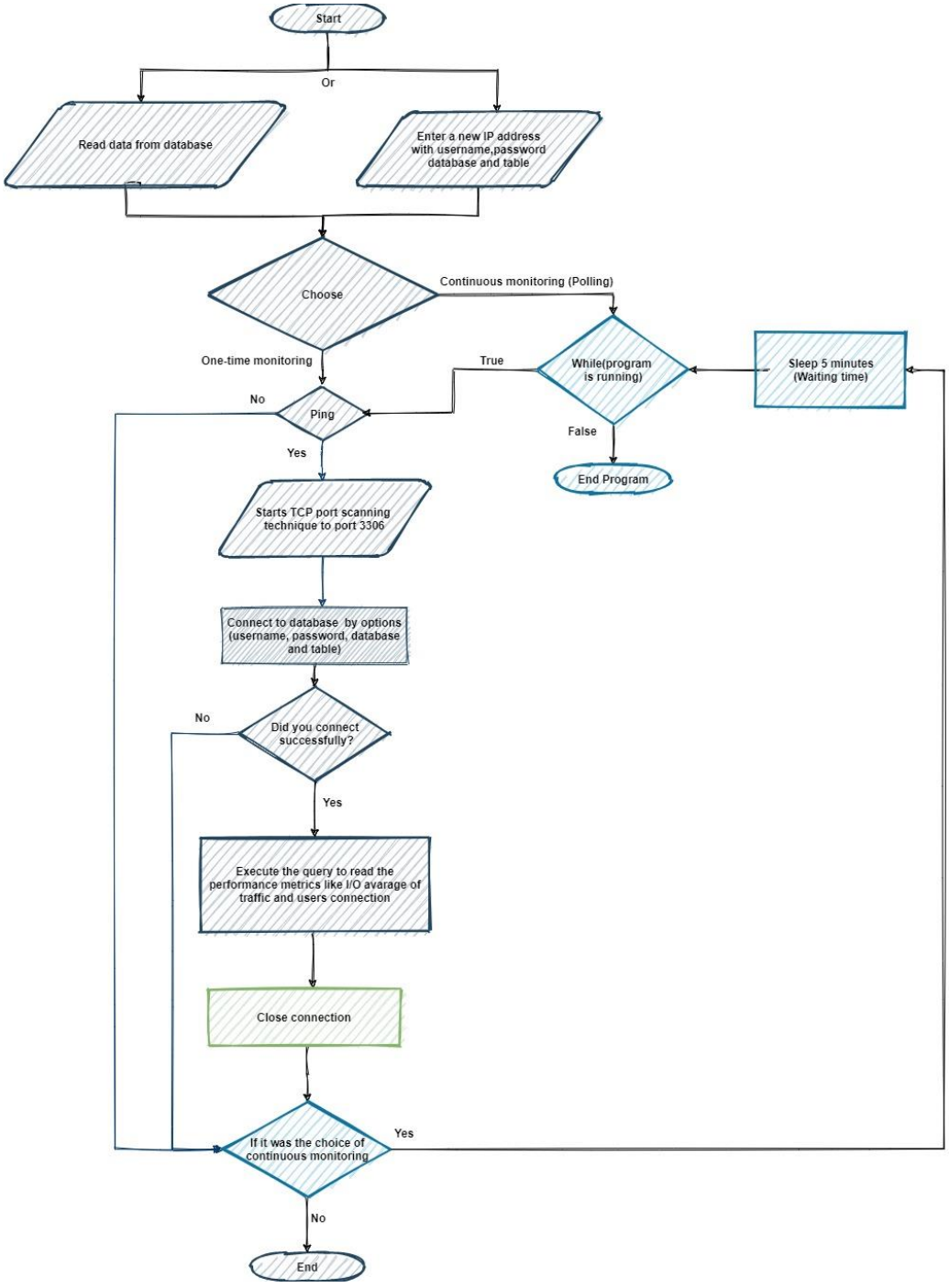


Figure 3.20: Database monitoring flowchart

3.6.2.5 Application monitoring diagram

This flowchart explains the monitoring process of the application. Therefore, it gets the data from different applications such as email and files. At first, for the FTP we should connect to the device by using a username, password, and IP address. On the other hand, for the email, we should connect it by using TCP port 465 with verifying if the SSL is available or not. The completion of the program depends on it being one-time or continuous monitoring. Figure 3.21 shows all details.

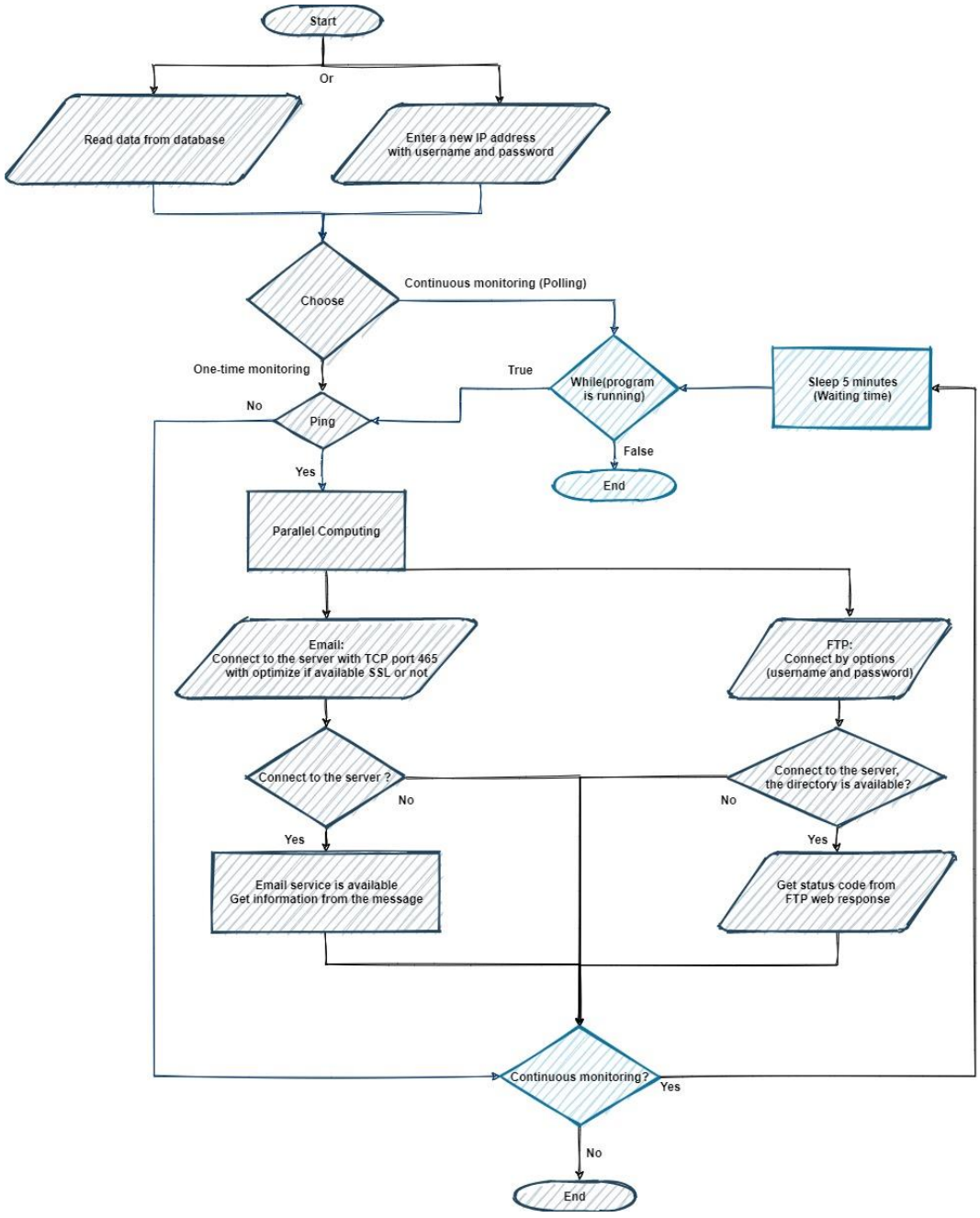


Figure 3.21: Application monitoring flowchart

3.6.2.6 Eventlog monitoring diagram

This flowchart explains the monitoring process of Eventlog. Therefore, it records information about important software or hardware events. At first, we should connect to the device by using a username and password, and IP address which is mandatory for WMI services. The program pulls the events according to the date entered by the network administrator and is categorized by type as in Table 3.1 so that the program can warn and display them graphically. The completion of the program depends on it being one-time or continuous monitoring. Figure 3.22 shows all details.

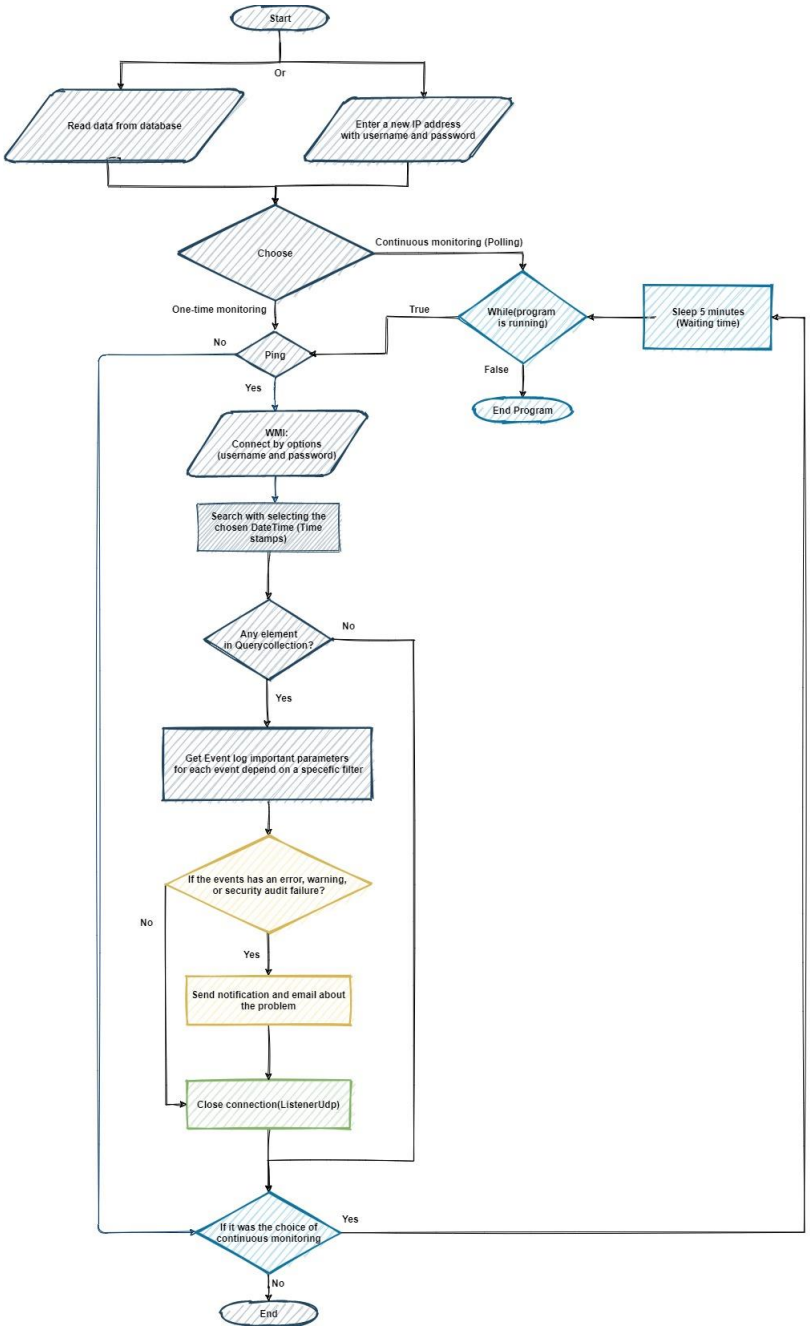


Figure 3.22: Eventlog monitoring flowchart

Table 3.1 shows the values of the Eventlog types, which helps us distinguish the important types for notification and pie charts.

Table 3.1: Eventlog types

Value	Meaning
1	Error
2	Warning
3	Information
4	Security Audit Success
5	Security Audit Failure

3.6.2.7 Cloud monitoring diagram

This flowchart explains the monitoring process of the cloud. Therefore, it gets the important parameter from the database in the cloud, which is (performance-schema). At first, we should connect to the server by using a username, password, IP address, and select the preferred SSL. We used a specific query to get the average for reading and writing operations, as well as the user information with the current and total connection. The completion of the program depends on it being one-time or continuous monitoring. Figure 3.23 shows all details.

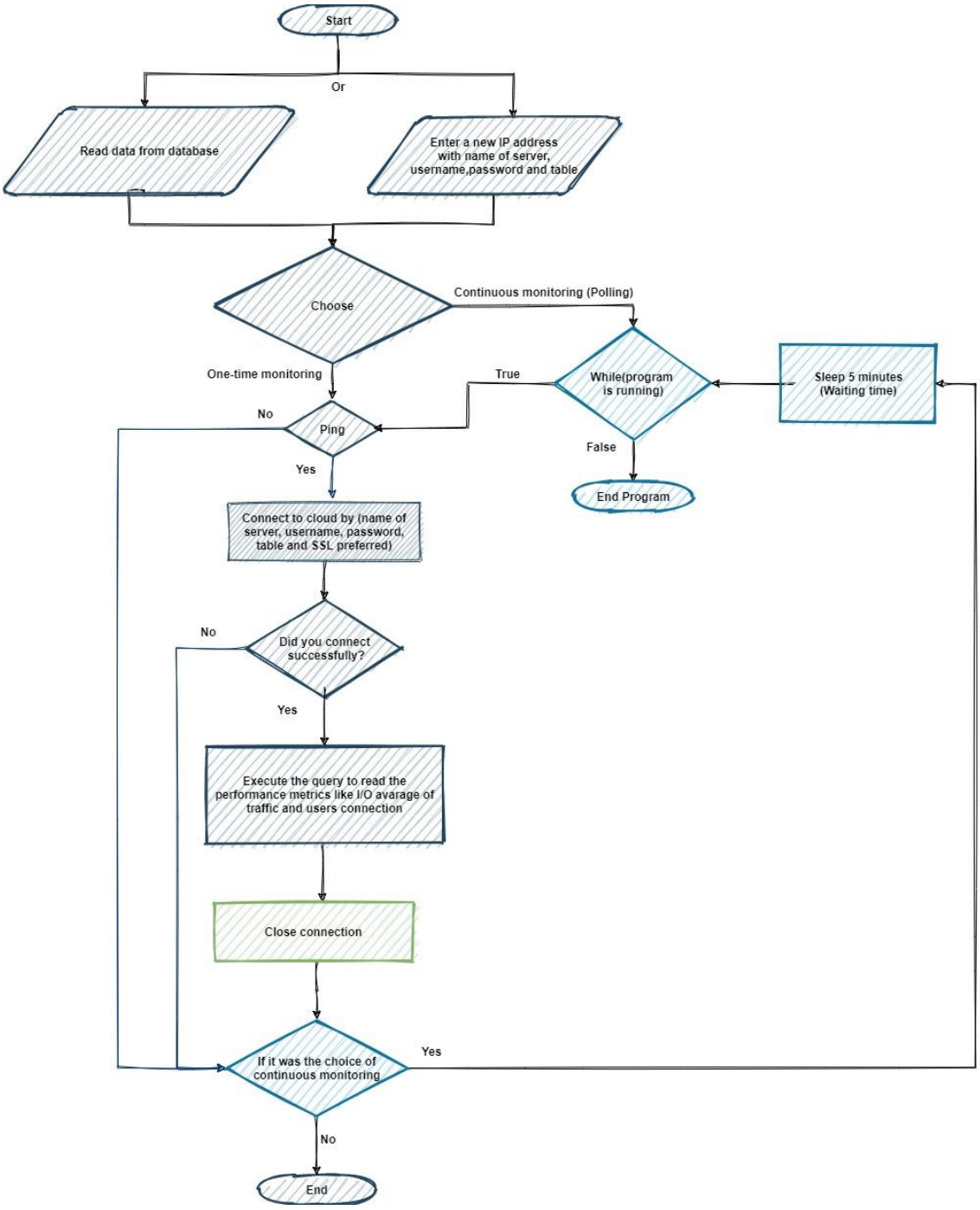


Figure 3.23: Cloud monitoring flowchart

3.6.2.8 SharePoint monitoring diagram

This flowchart explains the monitoring process of SharePoint. Therefore, it gets the important parameter from the server. At first, we should connect to the server by using a username, password, and link to the website. Moreover, you should log in to the Active Directory Federation Services (ADFS) for authentication. Afterthought, try to get the

performance metrics like health reports, the number of current page requests, and CPU usage. The completion of the program depends on it being one-time or continuous monitoring. Figure 3.24 shows all details.

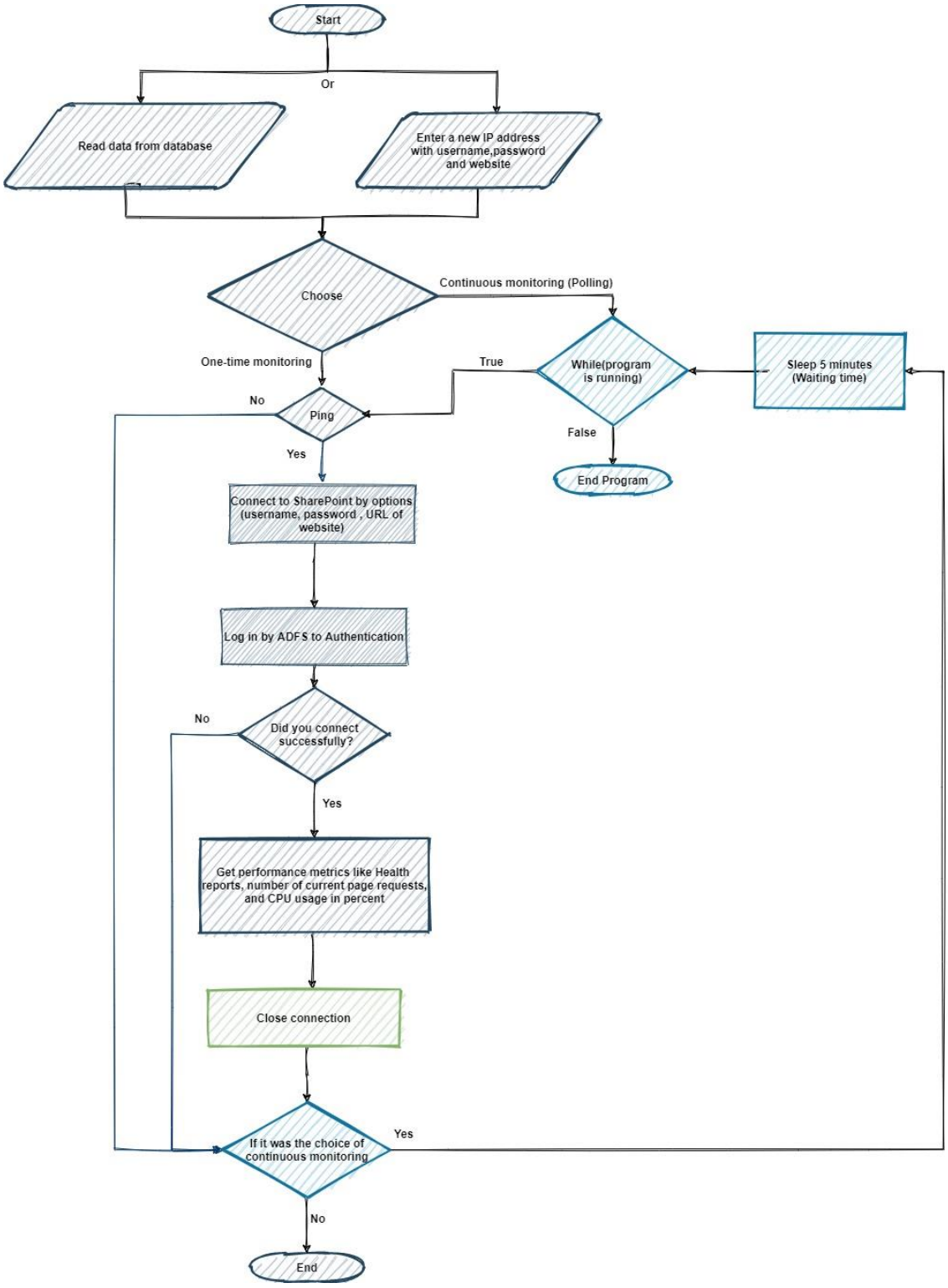


Figure 3.24: SharePoint flowchart

3.6.2.9 Citrix monitoring diagram

This flowchart explains the monitoring process of Citrix. Therefore, it gets the important parameter from the server. In the beginning, we should connect to the server by using a username, and password. Afterthought, try to get performance metrics like CPU, RAM usage, and downtime with the users' session information. Finally, the completion of the program depends on it being one-time or continuous monitoring. Figure 3.25 shows all details.

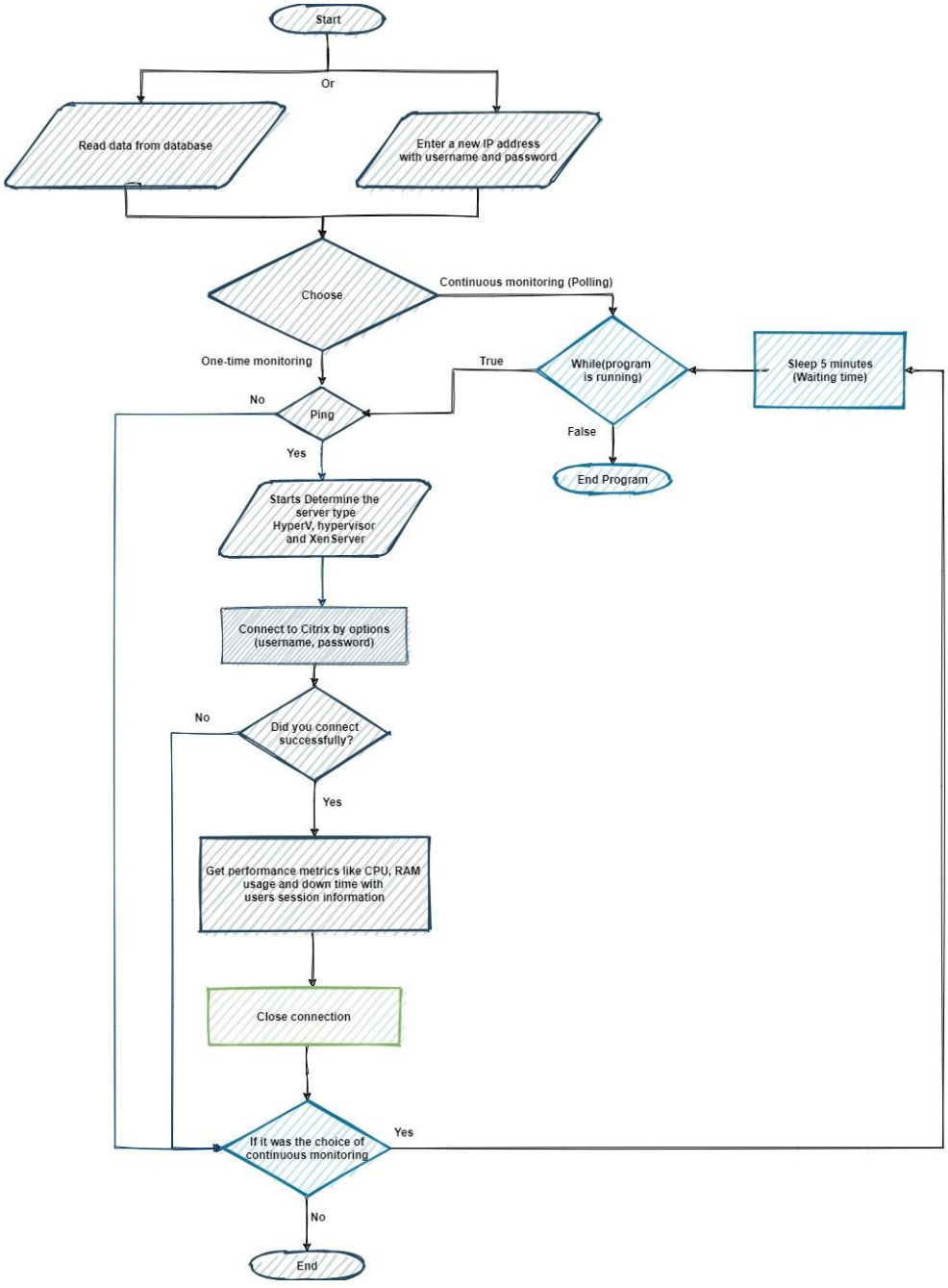


Figure 3.25: Citrix monitoring flowchart

3.7 System Design

3.7.1 Parallel programming

Our program can monitor in a real infrastructure like Taibah University. Thus, parallel programming will assist us to deal with many servers much faster and more efficiently. Indeed, parallel applications mainly use threads and shared memory, whether through libraries like threads or multithreaded languages like C# [22].

The challenge is parallel programming, which is the collision in displaying the data visually on the GUI and as shown by Figure 3.26 where there is more than one thread destined for the same component such as list view and pie chart, the simple solution to the process by queue technic. So, it will avoid UI thread and program freezing cause by cross-thread exceptions.

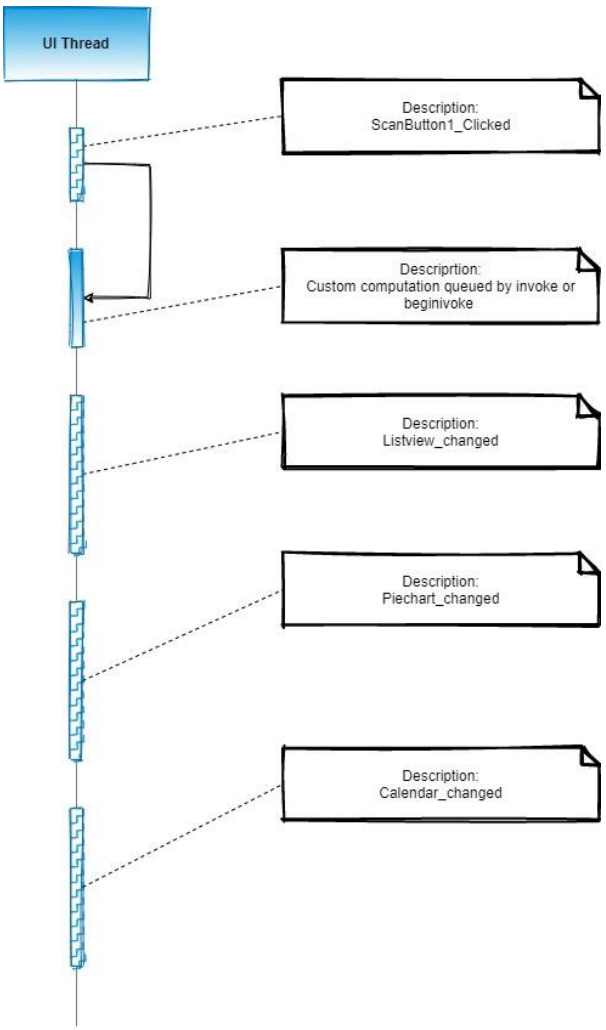


Figure 3.26: The UI Thread

The solution that we applied is to separate the processing operations from WMI and SNMP in parallel programming, at that time we applied the values. Furthermore, the values will be pulled in the background through the invoke methods. And with this, we saved a lot of time. Look at Figure 3.27.

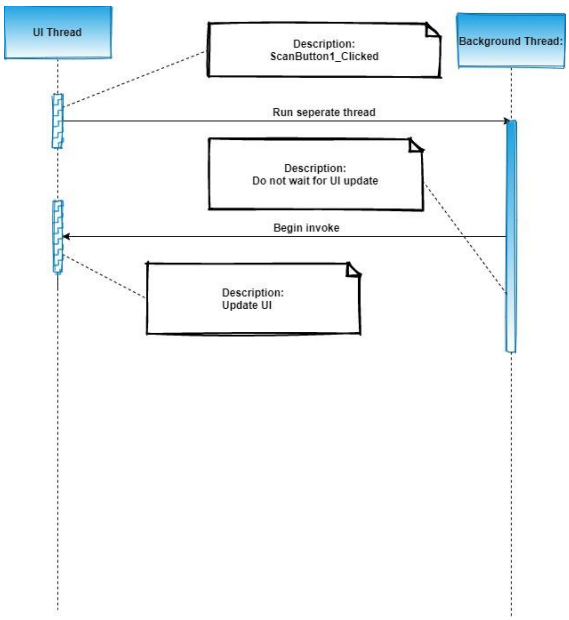


Figure 3.27: Invoke in UI thread

In Table 3.2 the time difference between the speed of software tools before and after parallel programming in the process of discovering any network element devices in a selected subnet. In addition to running the scan in the dashboard.
The way to prevent the application from freezing so that we can stop it, is to switch the method to the async task.

Table 3.2: The results of our experience in parallel programming

Process	Before parallel code	After parallel code	Percentage difference
Discover	2.25 minute	0.22 minute	+ 164%
Dashboard	2.16 minute	0.35 minute	+ 144%

3.7.2 OOP UML Diagram

Complex systems need parallel programming patterns to perform multiple operations at the same time. We have investigated the emergence of several problems in our system, including the interference and overlapping between the value of the variables and the slowdown of the system due to not adopting Object-Oriented Programming (OOP) techniques, and therefore we divided the services into class. Our project principle after

this technique is Do not Repeat Yourself (DRY) is to reduce code duplication, by collecting common methods in one parent class and reusing them instead of duplicating them. Figure 3.28 illustrates the flow in our program.

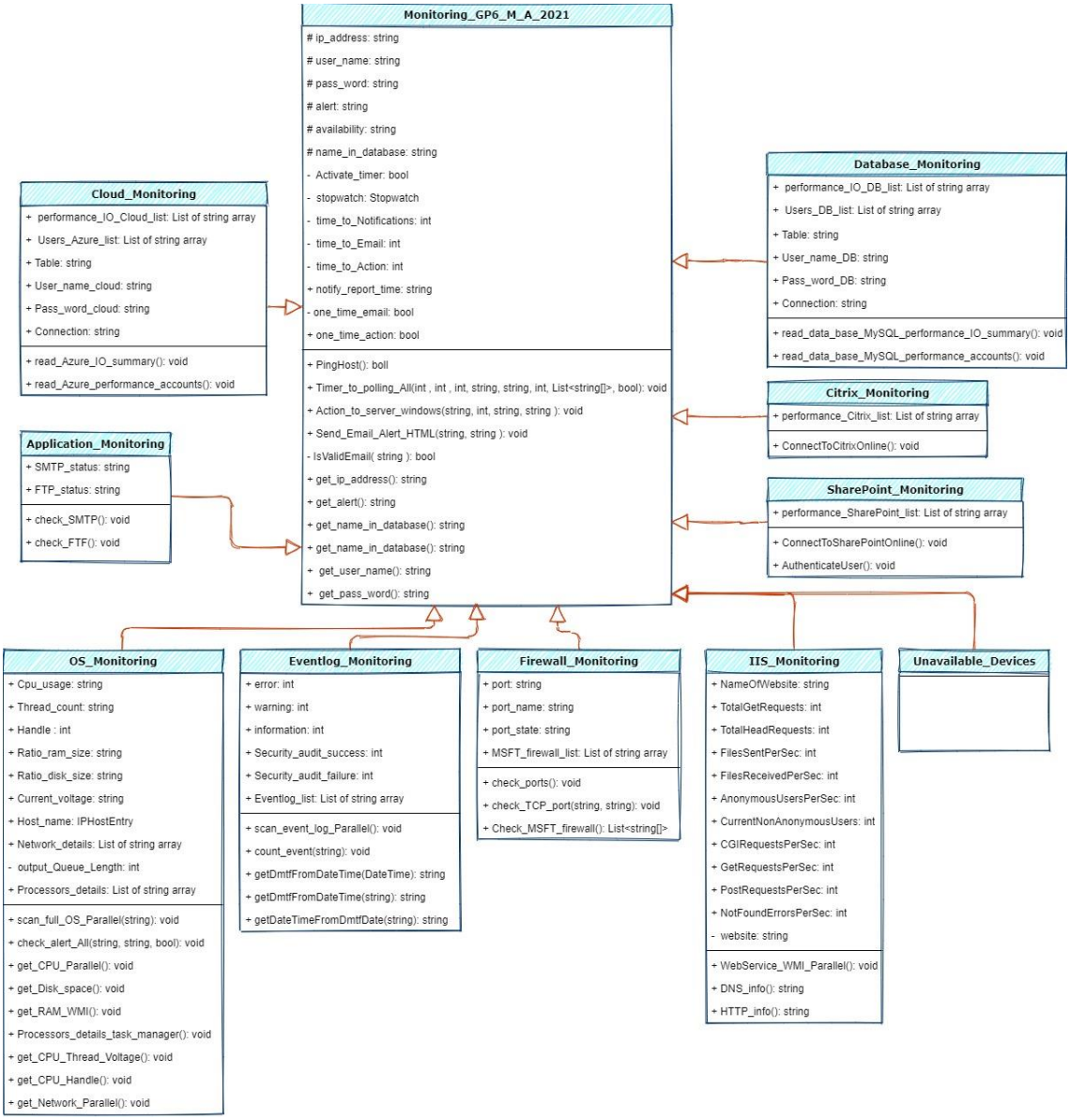


Figure 3.28: UML diagrams for OOP

3.7.3 Database ER-Diagram

3.7.3.1 Tables Relationships

The ER diagram is composed of 13 tables, 11 of them are for storing the different types of users, notification messages, Firewall, OS, Eventlog, Database, Cloud, Citrix, SharePoint, Application, and IIS monitoring. The remaining 2 are for collecting data from

services, and notifications logs. Figure 3.29 shows the relation between tables. In the following section, a description of each table is given.

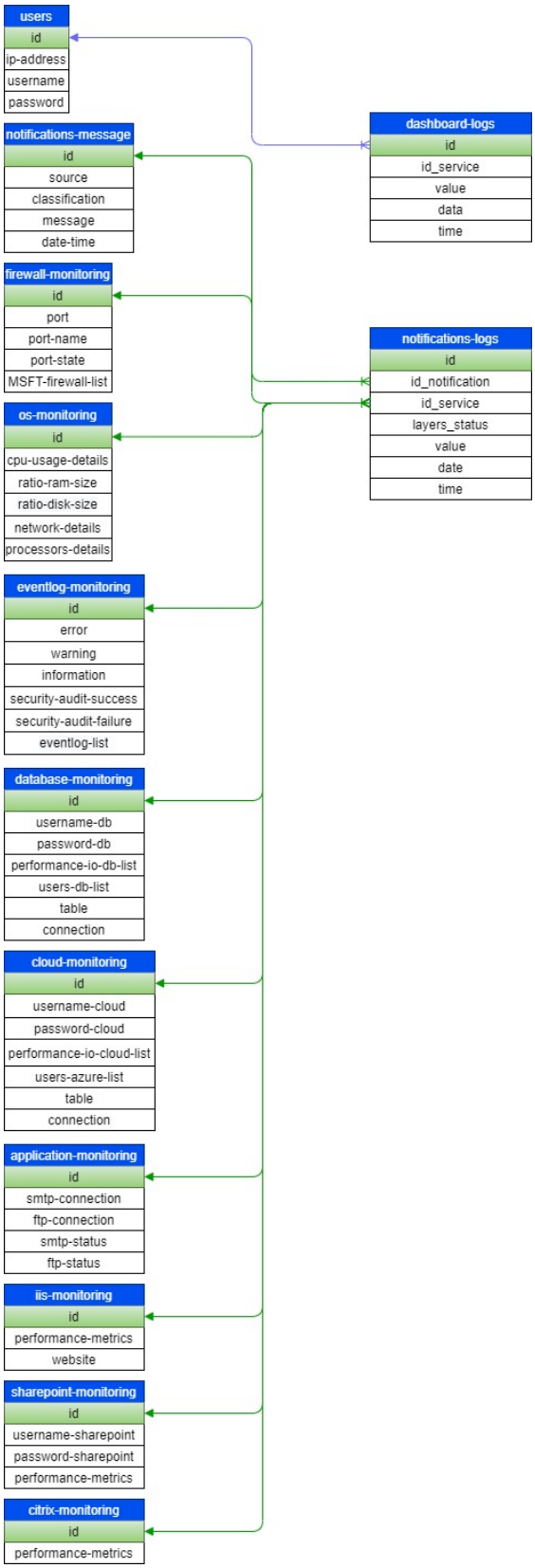


Figure 3.29: Database ER Diagram

3.7.3.2 Tables descriptions

- Table structure for table users

In Table 3.3, stores all authentication of users and password for each server used in this project. In Table 3.4, shows an example of data that are stored in this table.

Table 3.3: Users table structure

Column	Type	Null
id	int(10)	No
ip-address	varchar(100)	No
username	varchar(100)	No
password	varchar(100)	No

Dumping data for the users' table

Table 3.4: Example of users table

ID	IP address	Username	Password
2	192.168.1.90	Administrator	Abcd@1234

- Table structure for OS monitoring

In Table 3.5, stores all important parameters for each server used in this project. In Table 3.6 shows an example of data that are stored in this table.

Table 3.5: OS table structure

Column	Type	Null
id	int(100)	No
cpu-usage	varchar(3)	No
ratio-ram-size	varchar(3)	No
ratio-disk-size	varchar(3)	No

Dumping data for OS table

Table 3.6: Example of OS table

ID	CPU usage	RAM size	Disk size
2	80	90	20

- Table structure for table application

In Table 3.7, stores all important parameters for each server used in this project. In Table 3.8 shows an example of data that are stored in this table.

Table 3.7: Application table structure

Column	Type	Null
id	int(100)	No
smtp-connection	varchar(20)	No
ftp-connection	varchar(20)	No
smtp-status	varchar(20)	No
ftp-status	varchar(20)	No

Dumping data for application table

Table 3.8: Example of Application table

ID	SMTP connection	FTP connection	SMTP status	FTP status
2	Available	Available	EHLO SMTP	PathnameCreated

- Table structure for IIS monitoring

In Table 3.9, stores all important parameters for each server used in this project. In Table 3.10 shows an example of data that are stored in this table.

Table 3.9: IIS table structure

Column	Type	Null
id	int(100)	No
performance-metrics	varchar(20)	No
website-metrics	varchar(20)	No

Example of dumping data for IIS table, the values calculated per second

Table 3.10: Example of IIS table

ID	Total requests	Files sent	Anonymous users	CGI requests	Not found errors
2	30	4	2	1	5

3.7.4 Sequence Diagram

Detecting logic problems during an operation early, as well as documentation is one of the benefits that sequence diagrams design for us. In Figure 3.30 we can see the program processing when requesting information and communicating with a server.

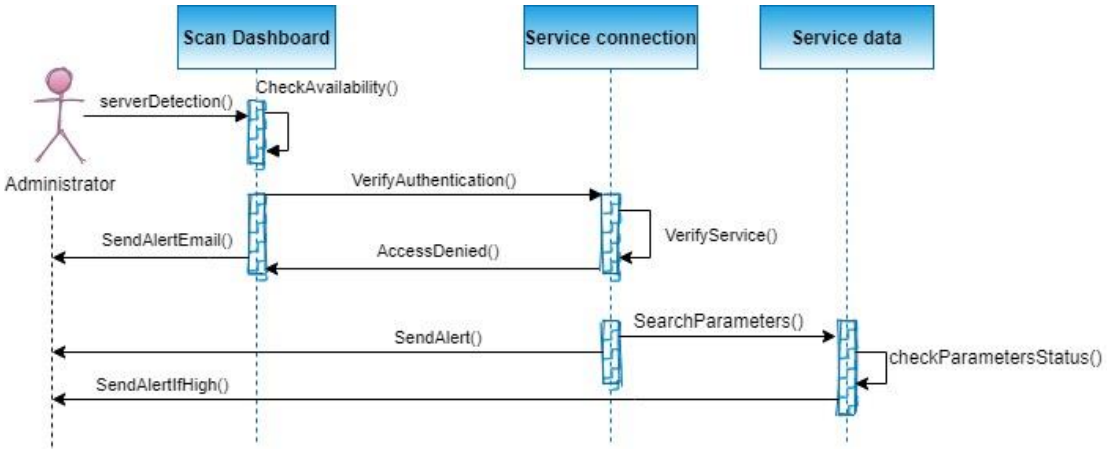


Figure 3.30: Sequence Diagram to monitor service

Figure 3.31 shows us the sequence of notifications so that if the consumption of resources on the server increases due to service, notifications will be sent to the device five minutes later. In addition, if the server remains in a high utilization state, it will send an email containing the problem, and then it will be forced to do an action, which includes a reboot of the device. Actually, repairing problems with our tool is distinguished from other technologies as it includes the feature of sending a signal and performing the server.

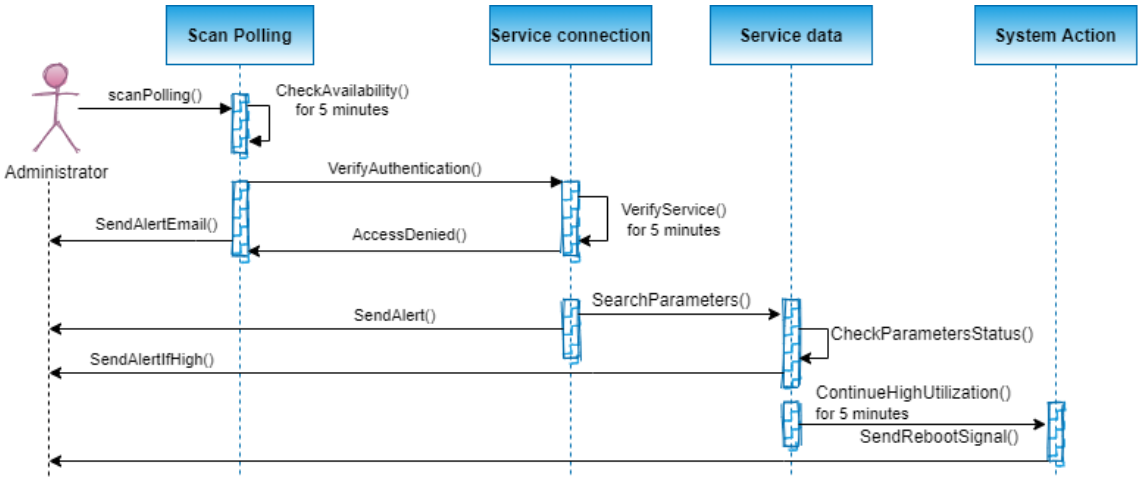


Figure 3.31: Sequence Diagram to the notification system

3.7.5 Notification management Design

The most important part of the monitoring system is the notification system for reporting problems to the network administrator or the NOC team. In Table 3.11, we notice the interaction during the time from the beginning of the problem arising to sending notifications to the device to the network administrator. After that, if the problem is not resolved, an email is sent with the details and then tries to solve the problem using different methods, including reboot.

Table 3.11: Notification interaction

Time after a problem arises	Notifications on the device	Send an email	Take action to solve a problem
5 minutes	Yes	-	-
10 minutes	-	Yes	-
15 minutes	-	-	Depends on the mode

3.7.6 File management Design

File storage has several benefits, including storing contact data within devices in case of losing the communication with the database, and reviewing monitoring of the network for a certain period of time. Also, this information is later used to build an Artificial Intelligence (AI) model to solve the network problems. In Table 3.12, the categories of

files saved by the program are shown, as well as an example of the file name, including system log storage in general, Eventlog file, and web file logs for sent files.

Table 3.12: Examples of log files

Function	File classification	Example of an auto-generated name
Dataset of IP address	Data	Dataset_IP
Eventlogs of servers	Data	Event_log
The email sent about the problem	Webpage	index_sent04-24-2021-11-37-AM

3.7.7 Proposed System Design

The proposed system design that is shown in Figure 3.32, shows the different components used to implement the network monitoring tool. Furthermore, it shows that the system is composed of three nodes systems that are connected to the monitoring tool. The first node has security that is consists of the NOC team and the administrator. The second node system has the service. And the last one is connected to the telecom towers which help to make all the buildings to be connected on one network. Thus, it helps to monitor them.

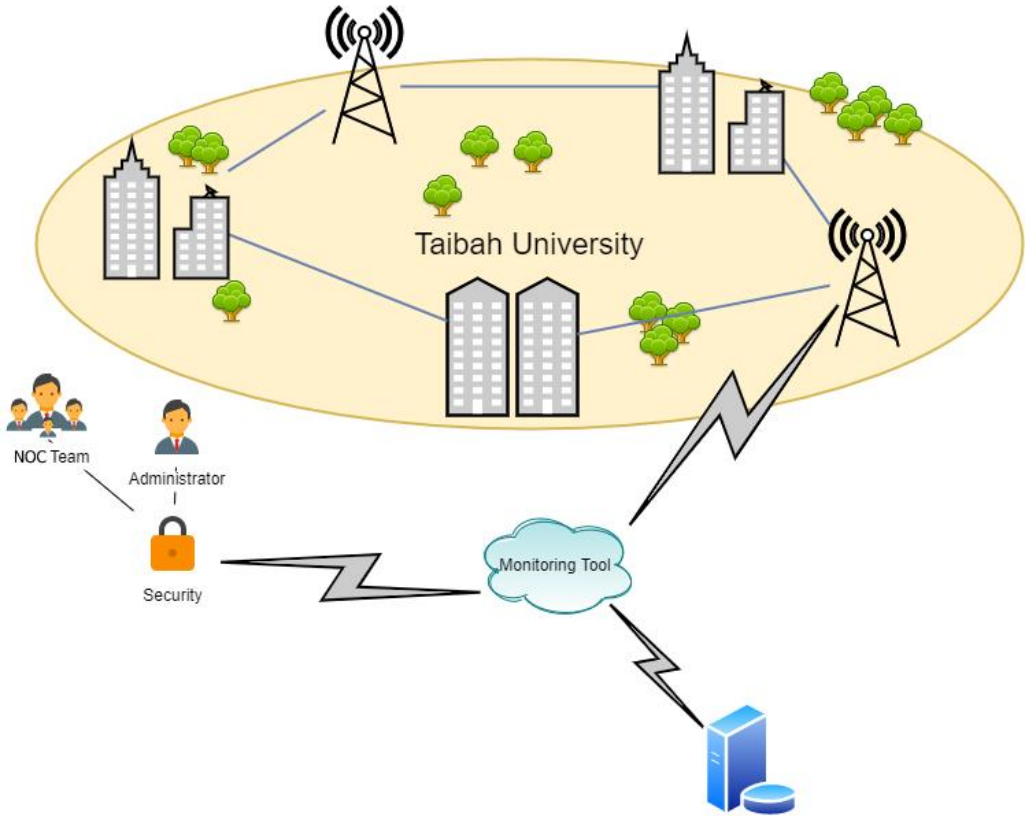


Figure 3.32: The Proposed System

3.7.8 User Interface (UI) Design

The user interface was developed by using visual studio and programmed by C# language. We intended to make the user interface as simple as possible so it would be simpler to use. Thus, we made a login screen that will open the main UI screen like in Figure 3.33. Furthermore, the UI window has an information window that shows the All Nodes window, Alerts window, Map window, Hardware Health Overview window, OS Overview, IIS overview. Moreover, it was mandatory to make the services so easy to access them by clicking one of the boxes on the left side of the user interface. As in Figure 3.34.

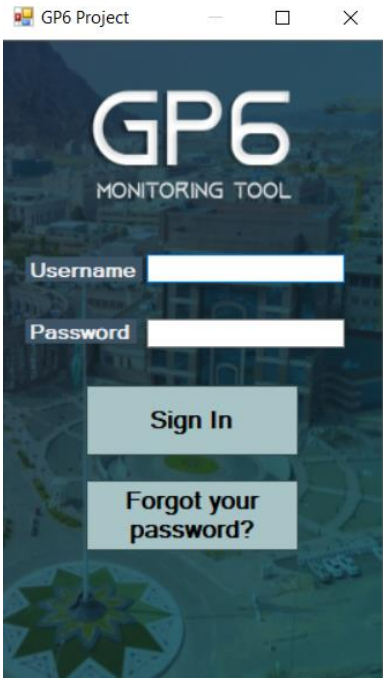


Figure 3.33: Login screen

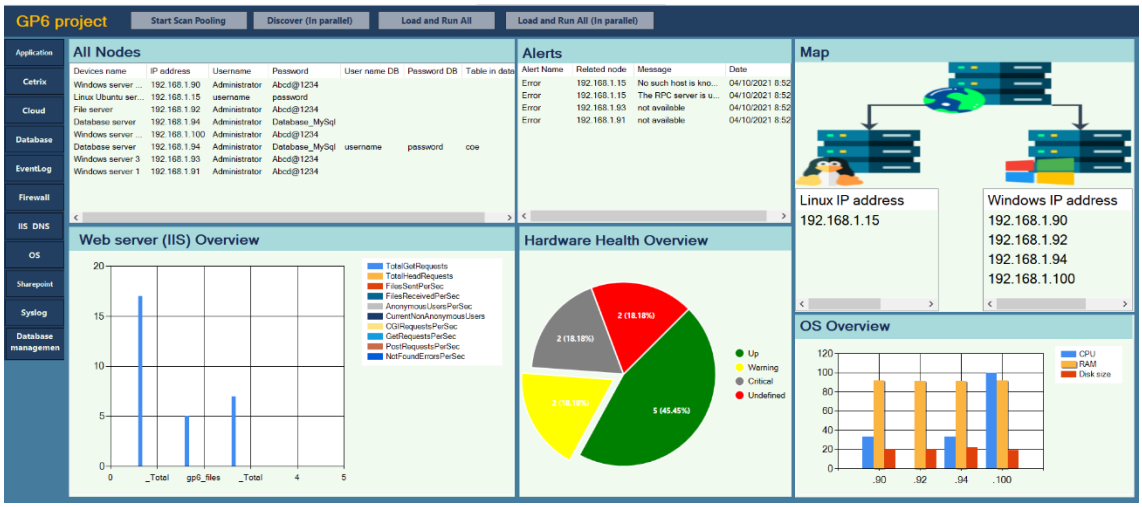


Figure 3.34: Dashboard screen

IMPLEMENTATION

4.1 Introduction

In this Chapter, we will be discussing our implementation process this semester. Among the things that we have done are: We have written the codes for the program, and have compiled the services that we will monitor, and we have created a graphical user interface for the program as well. In the coming sub-sections, we will explain this process and what happened in it.

This chapter was possible thanks to the help of the Deanship of Information Technology and Eng. Maher Ezzi, who allowed us to experience the services on a real university network, and this helped us to develop our monitoring tool better.

4.2 Early tests with switch

From the beginning of the first weeks in project 1, we took care of our practical Application, as this passed through several stages. Starting by using a switch with other devices to check the availability by using ping in a command prompt. However, it was necessary before that to assign IP address for static value not for a dynamic by default options, the steps in Figure 4.1.

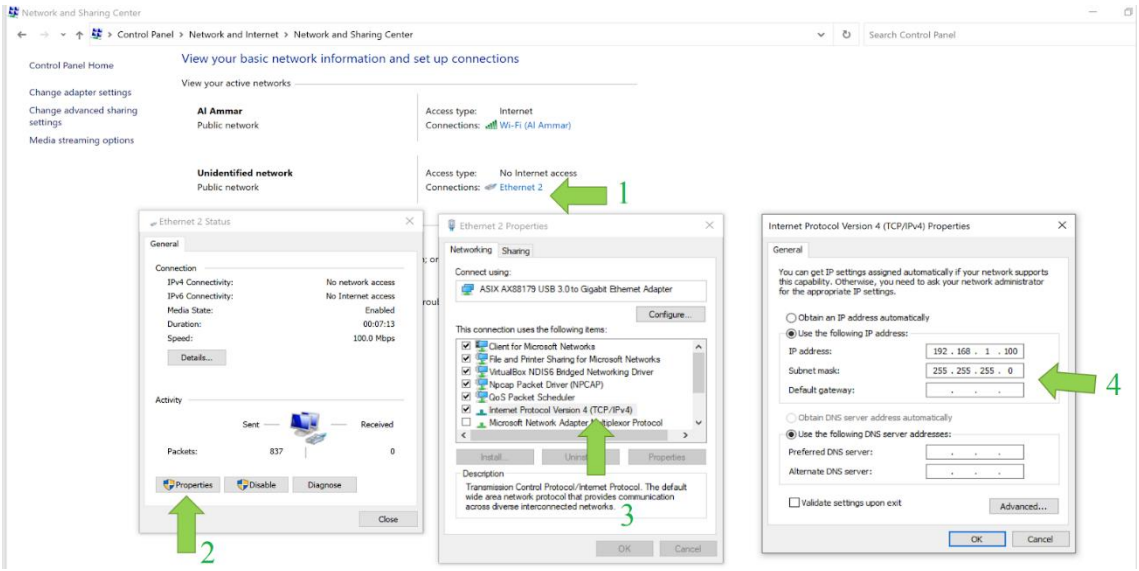


Figure 4.1: Steps to assign an IP address in windows

Based on the aforementioned in the last section. we start programming with something simple using the C# language and through Visual Studio Code that will modify and run the code by using the Mono program. Appendix (A) shows a simple program that uses a network information sub-library from System Net namespace, which provides all the

necessary functions for C# programs [23], and to check the device is available by using some functions to do an action by sending a packet and comparing with the replay status, Figure 4.2 shows the connection between devices and the use of a switch TP-link with a model TL-SF1005D.

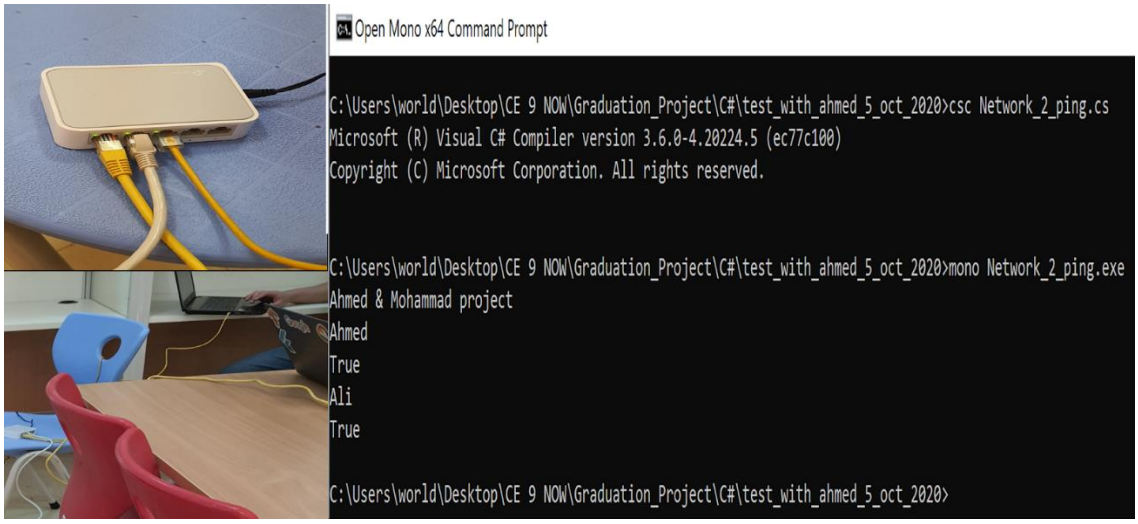


Figure 4.2: Program result with photos in the central university library

4.3 Virtual machine implementation

As can be seen from the report, several services have been added to the network monitoring tool, and a graphical user interface has been developed for it as well. With this, we have finished building the monitoring tool. In the beginning, we tried the tool on several servers on different virtual systems to make sure that we got a flexible experience and to ensure its success before trying it on real infrastructure. Some of the systems we tried are Windows server 2019, Linux Ubuntu server 18.04.4 LTS.

4.3.1 Early tests with socket programming with VMs

To try more features of a variety of systems that provide us with the ability to smoothly and easily to using VM in the device instead of connecting, we have used the oracle VM VirtualBox program for the test, with different systems like Windows, Linux, and macOS like what shown in Figure 4.3. With socket programming, it must modify the appropriate network feature to a bridged adapter to communicate with the device, like a switch.

The goal of system diversity is to experiment with different conditions by searching for and installing systems in addition to related systems like Nagios.

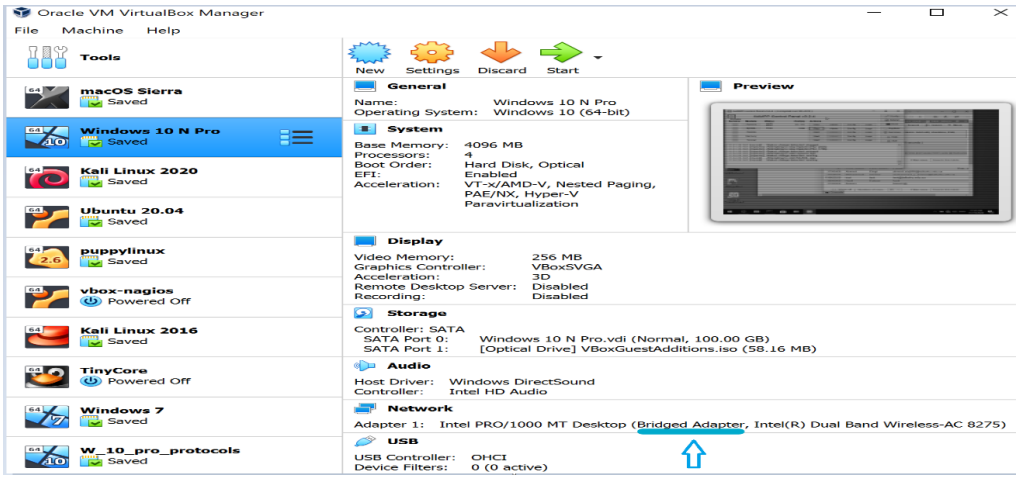


Figure 4.3: GUI of virtual box

Our project is interested in application services and databases that have been tried out by VM. We used the XAMPP program to make localhost with the MySQL database in Windows inside VM.

After this experience, we moved to Visual Studio 2019, and though we used a sub-library, which is MySQL Client. We create a table in the database and a table that contained some elements, and we activated the powers to access the database from outside the same system so that we could through the program located in appendix (A), and in Figure 4.4 the result where program verify the status of the devices and then communicate with the database by entering the necessary information which IP, username, password, and database name. Then connect to the database and read data through a loop of everything in the rows.

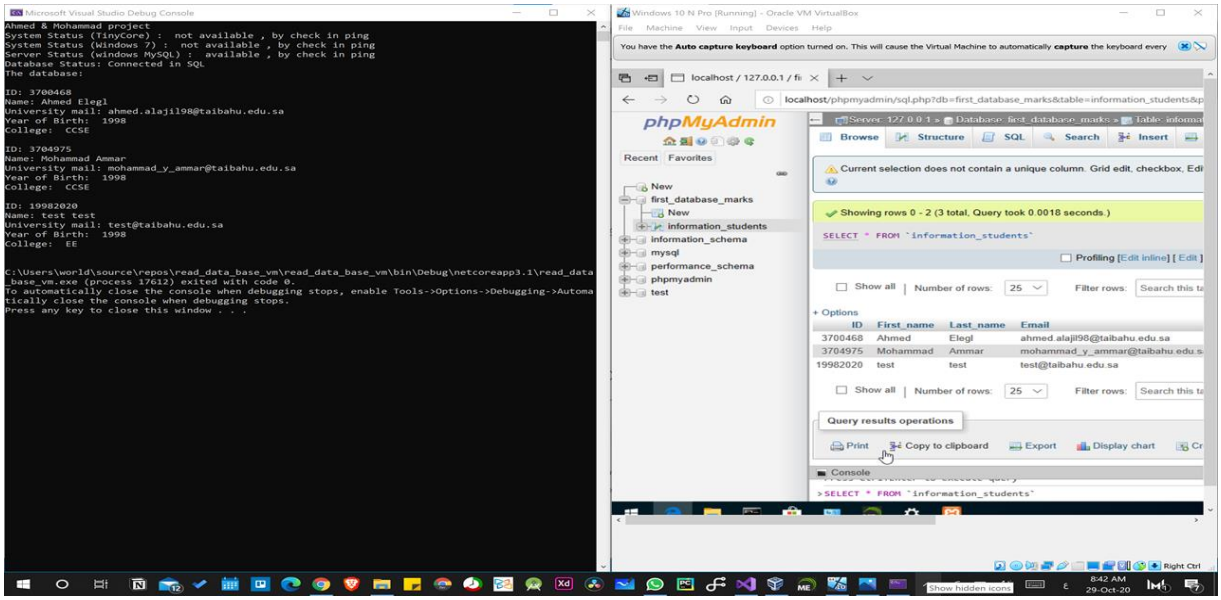


Figure 4.4: Program result with the MySQL database

4.3.2 Early tests with socket programming with GUI

To make the program easier to use, a graphical interface must be made. We have implemented a program that contains some of the stuff that was mentioned in the project, from checking services with devices and polling method, where the program checks if the device stops, it alerts through notifications to the device and if it increases more than once, it sends an email automatically about the problem and its details like what shown in Appendix (B), the program includes these features like in Figure 4.5, it is possible to add devices from the main, check and request more information, for example: reading information by WMI, read from a database, and buttons to open another window forms.

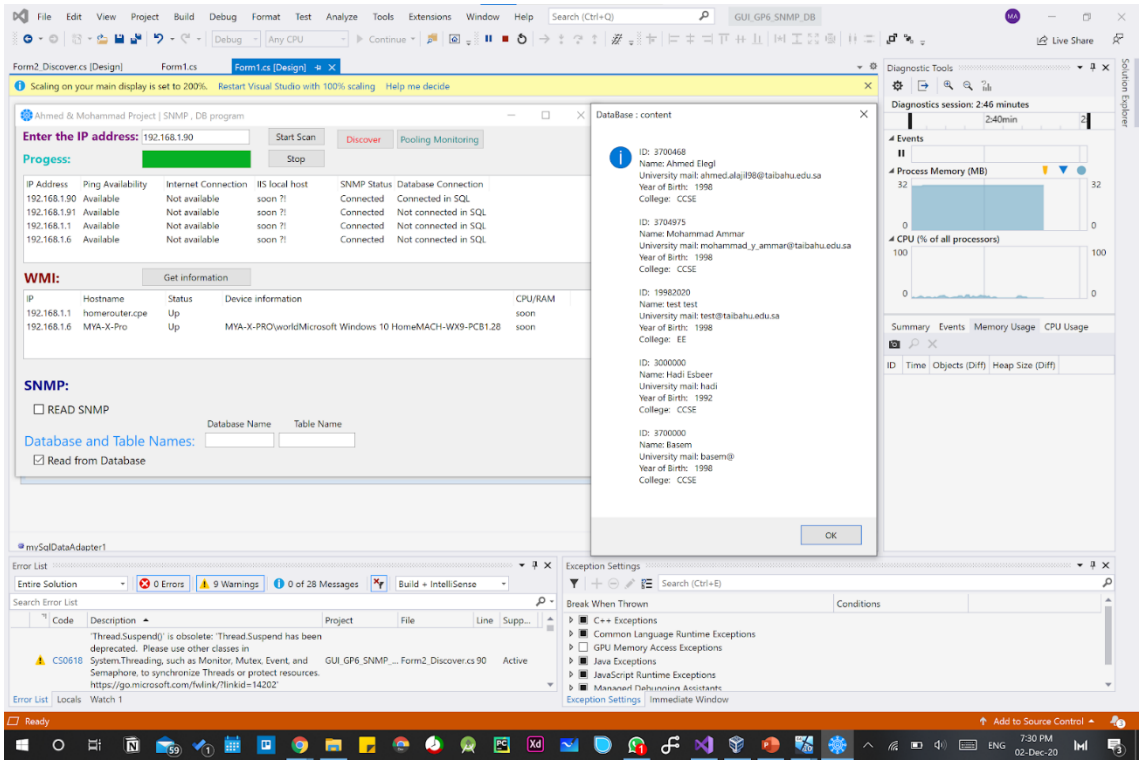
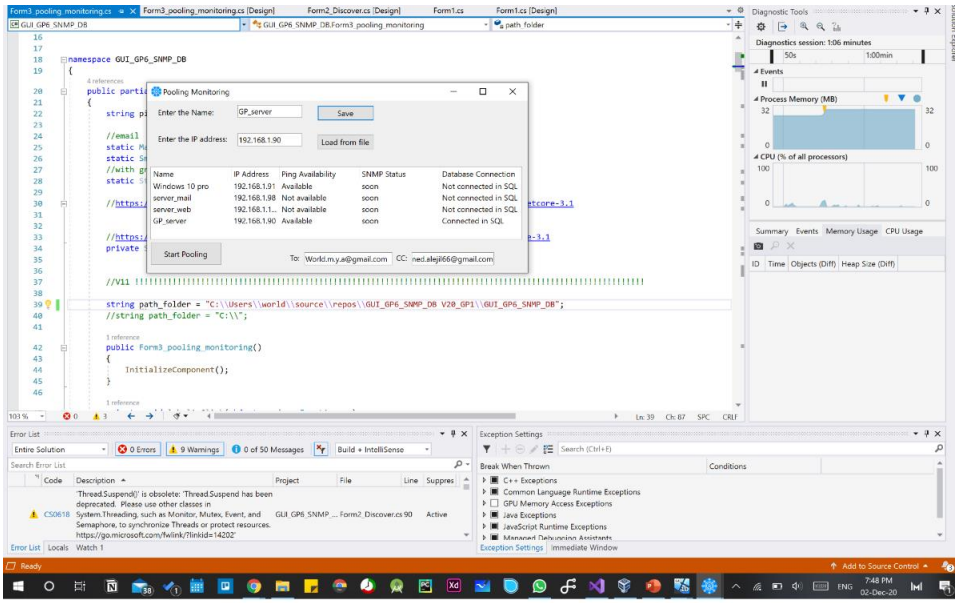


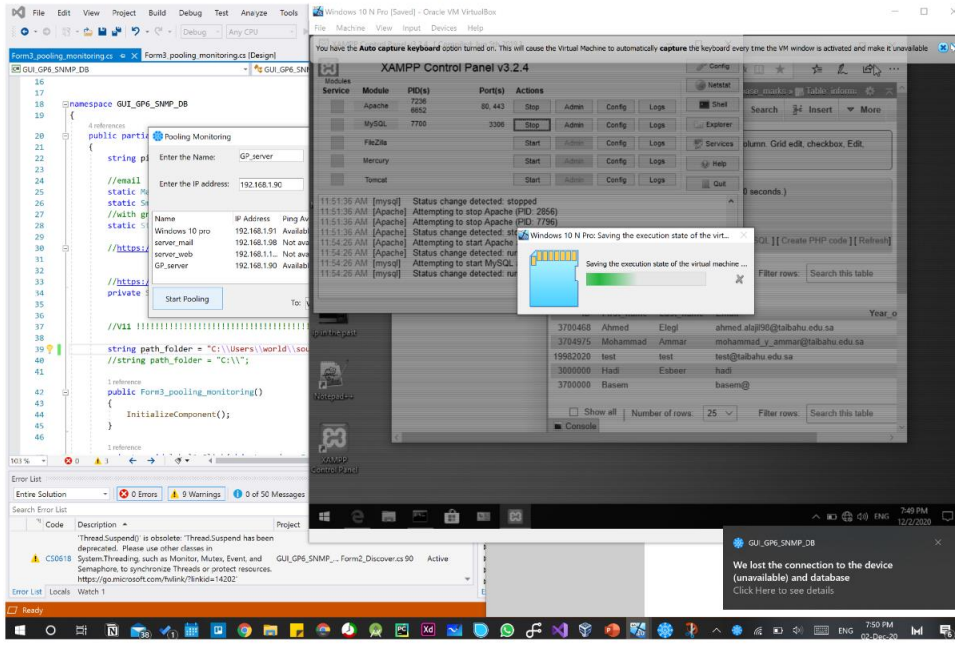
Figure 4.5: First Form in GUI

When you open the polling monitoring window, the program automatically loads an existing database and examines it directly, as in Figure 4.6 in part (A), and it is possible to add new devices with them.

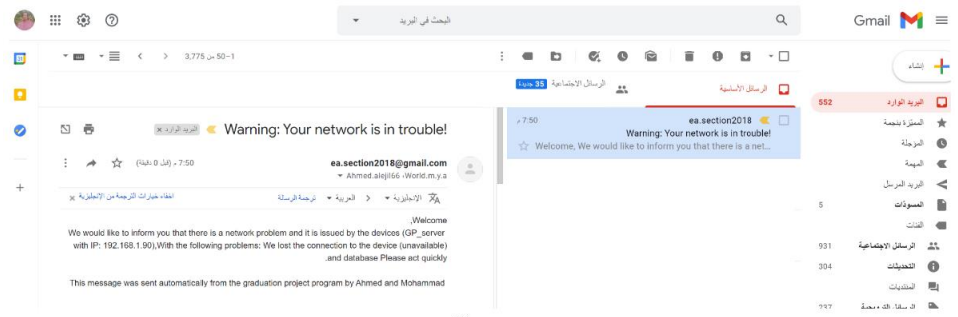
We ought to try the NMS behavior, like the polling monitoring method, the program will continuously check every five seconds in terms of its availability and the database connection will be alerted directly in a notification to the device as in part (B) the error and its description, and then after displaying five notices, the system will automatically send a message to the entered email describing the problem and from which any device it originating from as in part (C) of the figure.



A



B



C

Figure 4.6: Polling with alert and send email

4.3.3 Login screen

The login screen is designed using visual studio, it has the username and password which is necessary to get into the network monitoring tool. Also, it has a "Forgot your password?" button which allows you to restore your account if you lost the password. Look at Figure 4.7.

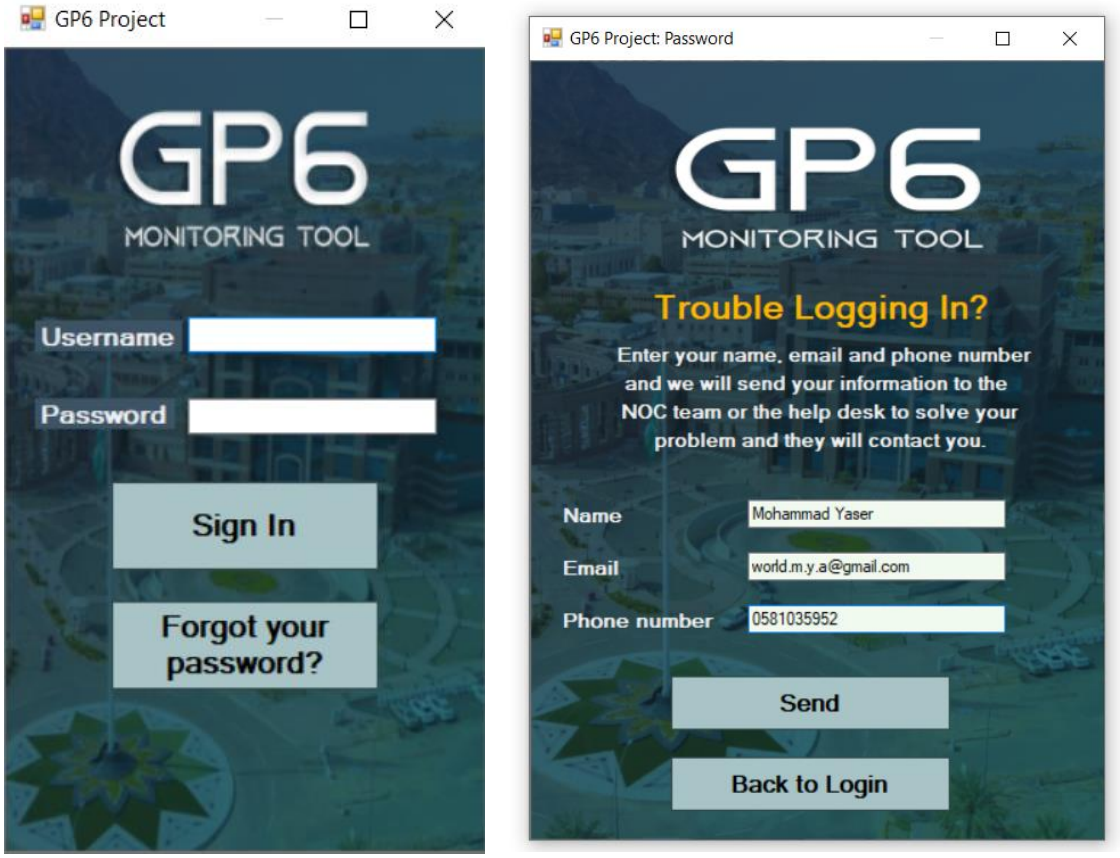


Figure4.7: Login and forgot tout password screen window

4.3.4 General Dashboard

The main user interface contains a summary of the most important services. As shown in Figure 4.8, all nodes include details about all servers like IP address, username and password, and other information. The graphical user interface shows many diagram figures in different formats, which provide a summary of the status of the webserver and details of the system resources such as processor, memory, and storage, in addition to a simple map showing the classification of the system type from Windows or Linux.

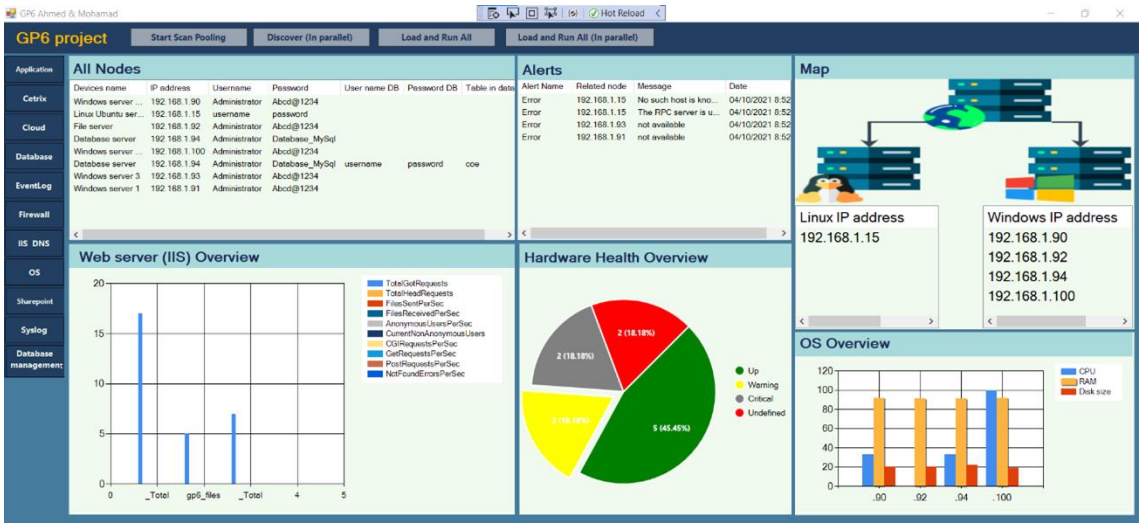


Figure 4.8: General Dashboard

4.3.5 Custom Dashboard

The network administrator can access a single server to display all its details in one place, and this is the role of the custom dashboard as it clarifies the main services for each server such as its warnings, electricity status, uptime, operating system performance data, and task manager as in Figure 4.9.

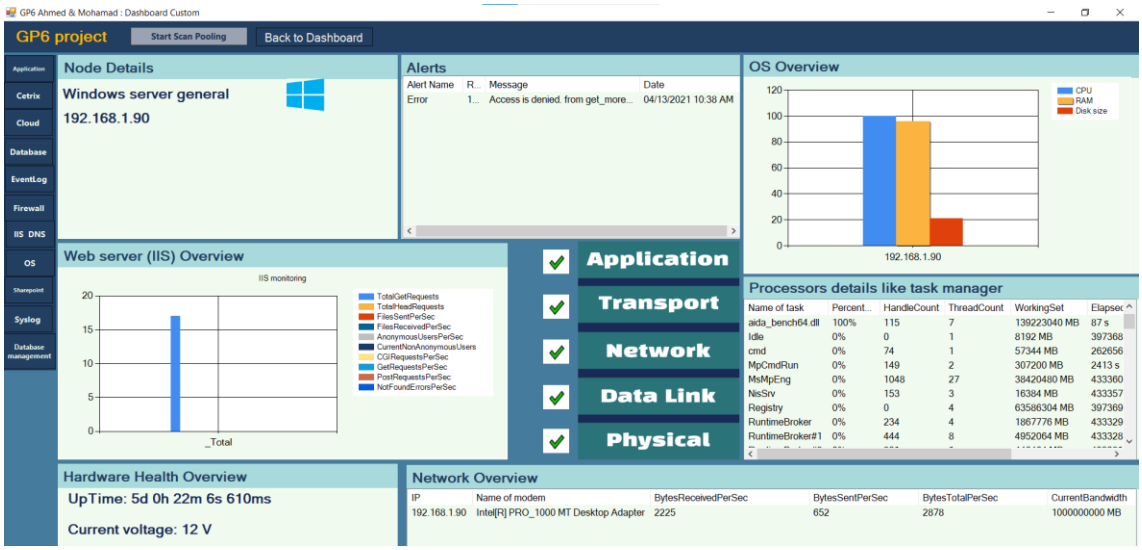


Figure 4.9: Custom Dashboard to windows server

As shown in Figure 4.10, we verify that the interface adapts the system icon to Linux, and it displays the working or non-working layers according to the status of the server, which helps effectively to know the problem and in which layer it is. As is the case when trying Linux, we notice that we have enabled the SNMP basis so that it only recognizes

the system type and the uptime. Therefore, all layers except the Application layer have an error due to their inability to access.

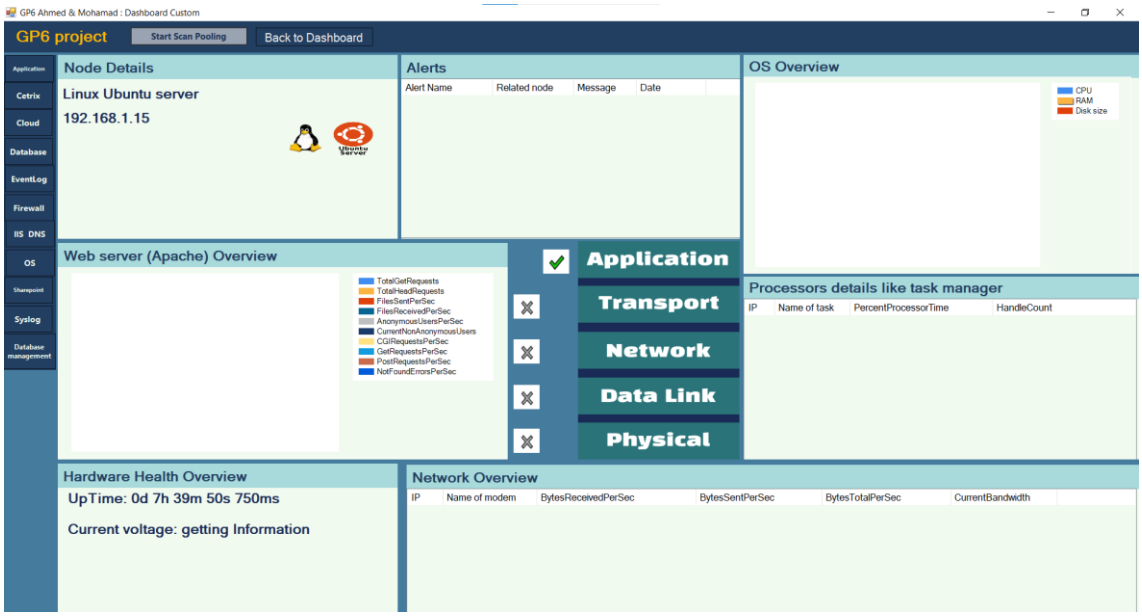


Figure 4.10: Custom Dashboard to Linux Ubuntu server

The custom dashboard can distinguish the services on the server. As was explained in the aforementioned server, a web server differentiates between Apache for Linux or IIS for Windows. In Figure 4.11, the database server was able to determine the database type, and the GUI was prepared accordingly with details of the connection information and the active users.

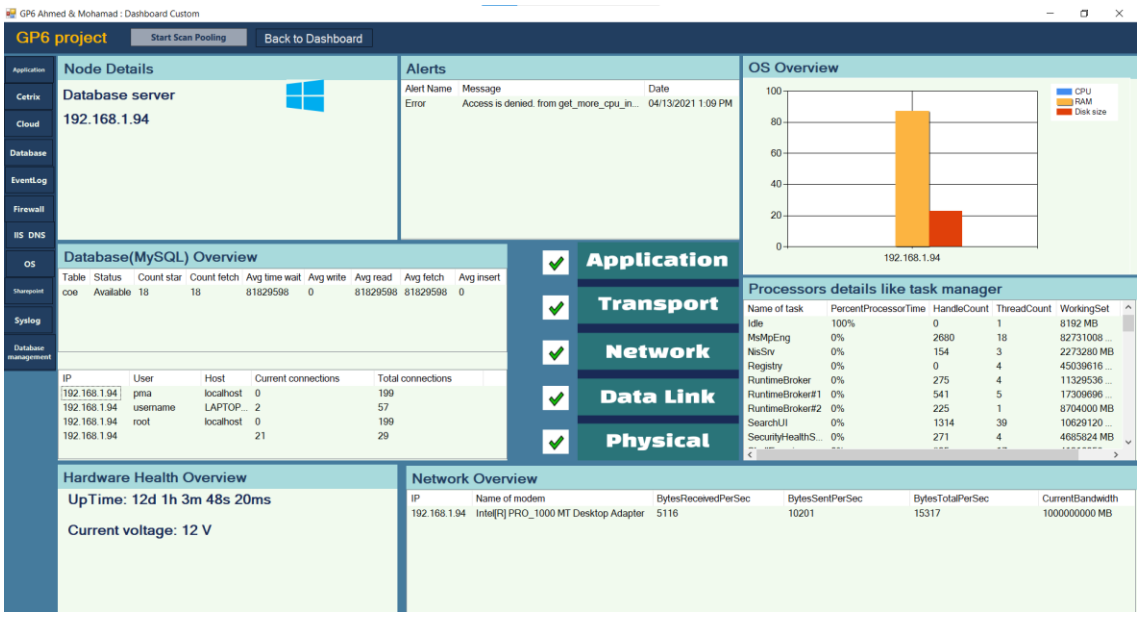


Figure 4.11: Custom Dashboard to database server

4.3.6 Database monitoring

We installed the MySQL database on a server. Later, we took several steps to activate the database (performance schema) which we will be able to read the information as in Figure 4.12. The result shows the traffic status of the server like the write and reads operations, the most active users, and the details for that or the error if there is that.

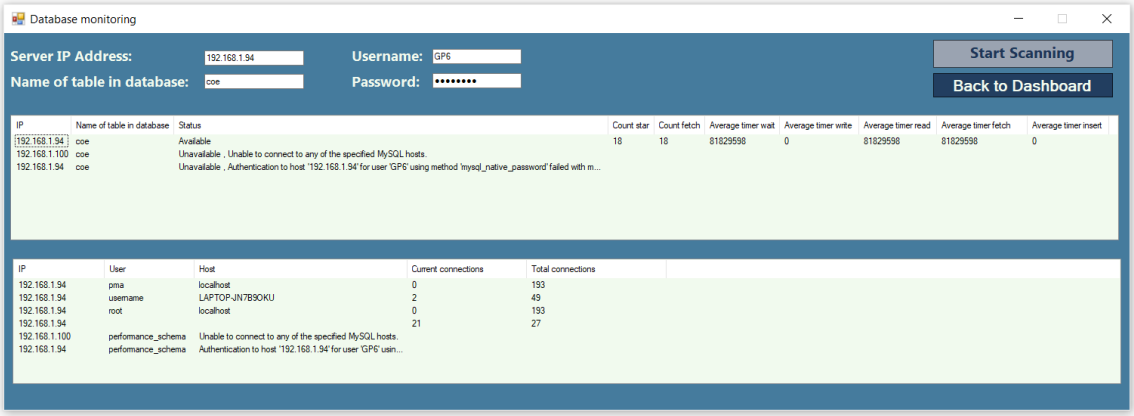


Figure 4.12: Custom Dashboard layers

The code for database monitoring is divided into two methods in a class, the first method in Figure 4.13 for traffic parameters like average time of reading and writing in a database, the second method in Figure 4.14, is similar to the first, but it is responsible for reading users and their actual traffic for the entire database, a sense that we do not need to enter a table like the first method. Of course, all these data store in a list of an array to show in any place in our programs like specific database service or custom dashboard.

```
public void read_data_base_MySQL_performance_io_summary()
{
    //configure database connection
    string name_data_base = "performance_schema";
    string connectionString = "datasource=" + ip_address + ";port=3306;username=" + User_name_DB + ";password=" + Pass_word_DB + ";database=" + name_data_base + "; CharSet=utf8;";
    // Query
    string query = "SELECT* FROM `table_io_waits_summary_by_table` WHERE `OBJECT_NAME` = '" + Table + "'";
    // Prepare the connection
    MySqlConnection databaseConnection = new MySqlConnection(connectionString);
    MySqlCommand commandDatabase = new MySqlCommand(query, databaseConnection);
    commandDatabase.CommandTimeout = 60;
    MySqlDataReader reader;
    try
    {
        // Open the database
        databaseConnection.Open();
        reader = commandDatabase.ExecuteReader();
        while (reader.Read())
        {
            string COUNT_STAR = reader["COUNT_STAR"].ToString();
            string COUNT_FETCH = reader["COUNT_FETCH"].ToString();
            string AVG_TIMER_WAIT = reader["AVG_TIMER_WAIT"].ToString();
            string AVG_TIMER_WRITE = reader["AVG_TIMER_WRITE"].ToString();
            string AVG_TIMER_READ = reader["AVG_TIMER_READ"].ToString();
            string AVG_TIMER_FETCH = reader["AVG_TIMER_FETCH"].ToString();
            string AVG_TIMER_INSERT = reader["AVG_TIMER_INSERT"].ToString();
            string[] info_database = new string[] {ip_address,Table,"Available", COUNT_STAR, COUNT_FETCH, AVG_TIMER_WAIT, AVG_TIMER_WRITE, AVG_TIMER_READ, AVG_TIMER_FETCH, AVG_TIMER_INSERT };
            performance_IO_DB_list.Add(info_database); // store to List
            Connection = "Available";
        }//while
    }//try
    catch (Exception ex)
    {
        alert = ex.Message;
        Connection = "Unavailable , " + ex.Message;
        string[] info_database = new string[] { ip_address, Table, Connection };
        performance_IO_DB_list.Add(info_database); // store to List
    }//catch
}//read read_data_base_MySQL_performance_io_waits_summary
```

Figure 4.13: Database I/O codes

```
public void read_data_base_MySQL_performance_accounts()
{
    //configure database connection
    string name_data_base = "performance_schema";
    string connectionString = "datasource=" + ip_address + ";port=3306;username=" + User_name_DB + ";password=" + Pass_word_DB + ";database=" + name_data_base + "; CharSet=utf8;";//CharSet=utf8
    // Query
    string query = "SELECT * FROM `accounts`";
    // Prepare the connection
    MySqlConnection databaseConnection = new MySqlConnection(connectionString);
    MySqlCommand commandDatabase = new MySqlCommand(query, databaseConnection);
    commandDatabase.CommandTimeout = 60;
    MySqlDataReader reader;//MYA
    try
    {
        // Open the database
        databaseConnection.Open();
        reader = commandDatabase.ExecuteReader();
        while (reader.Read())
        {
            string USER = reader["USER"].ToString();
            string HOST = reader["HOST"].ToString();
            string CURRENT_CONNECTIONS = reader["CURRENT_CONNECTIONS"].ToString();
            string TOTAL_CONNECTIONS = reader["TOTAL_CONNECTIONS"].ToString();
            string[] users_info = new string[] { ip_address, USER, HOST, CURRENT_CONNECTIONS, TOTAL_CONNECTIONS };
            Users_DB_list.Add(users_info);// store to List
        }//while
    }//try
    catch (Exception ex)
    {
        string[] users_info = new string[] { ip_address, name_data_base, ex.Message };
        alert = ex.Message + " From " + "read_data_base_MySQL_performance_io_waits_summary";
        Users_DB_list.Add(users_info);// store to List
    }//catch
}//read read_data_base_MySQL_performance_io_waits_summary
```

Figure 4.14: Database users codes

4.3.7 Application monitoring

The Applications included email and file services via protocols SMTP, FTP. Conversely that required setup in servers to enable that. In Figure 4.15. The result shows the Application status of the server like the FTP status, and the details for the SMTP server.

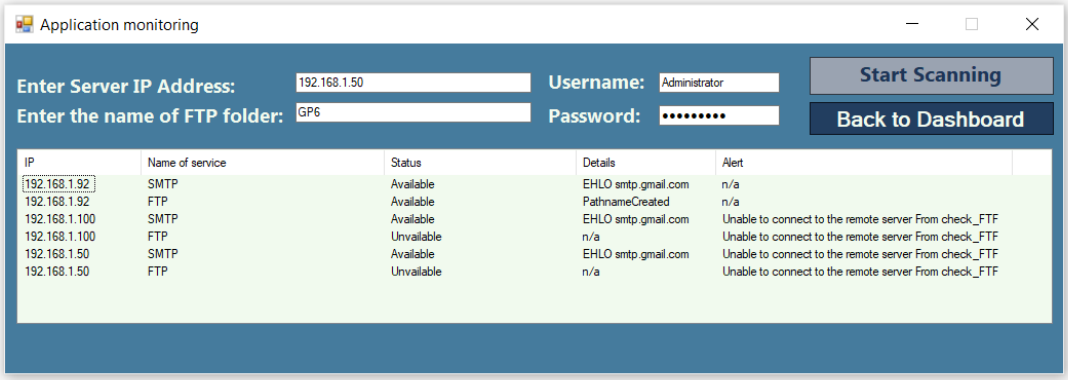


Figure 4.15: Application monitoring window

The code for Application monitoring is separated into two methods for SMTP and FTP, in Figure 4.16 the code to check if have access to FTP in the server and create a folder to check the access also.

```
public void check_FTF()
{
    try
    {
        var dt = DateTime.Now;
        string directory = dt.ToString("MM-dd-yyyy-h-mm-tt");
        WebRequest request = WebRequest.Create("ftp://" + ip_address + "/" + directory);

        request.Method = WebRequestMethods.Ftp.MakeDirectory;
        request.Credentials = new NetworkCredential(user_name, pass_word);
        using (var resp = (FtpWebResponse)request.GetResponse())
        {
            Console.WriteLine(resp.StatusCode);
            FTP_Connection = "Available FTP";//for debug
            FTP_status = resp.StatusCode.ToString();
        }//using
    }//try
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);//for debug
        alert = ex.Message + " From check_FTF";
        FTP_Connection = "Unvailable";
    }//cactch
}//check_FTF
```

Figure 4.16: FTP code

4.3.8 Firewall monitoring

Firewall results are based on the port availability. Moreover, it gets the information from WMI and directs it to the windows Firewall rule in the server. Look at Figure 4.17.

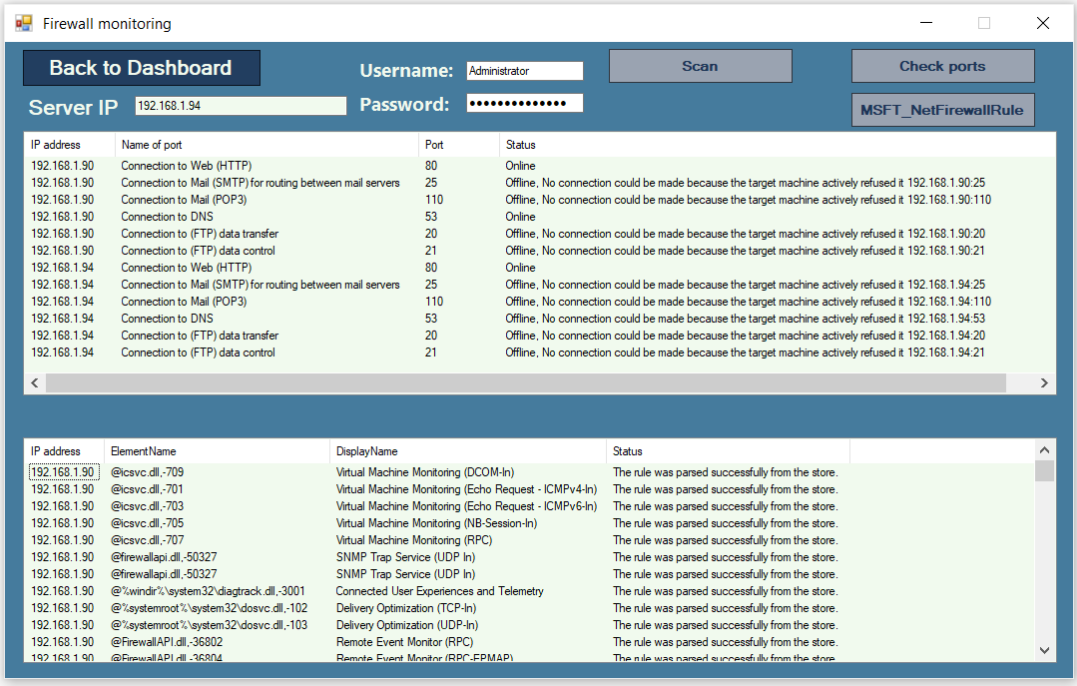


Figure 4.17: Firewall monitoring window

The code of the Firewall monitoring is separated into two methods that use the TCP port scanning technique as in Figure 4.18. furthermore, in Figure 4.19 the codes that are shown in Figure 4.19 are used for reading by using WMI.

```
public Firewall_Monitoring(string ip, string user, string pass)
{
    ip_address = ip;
    user_name = user;
    pass_word = pass;
}
2 references
public void check_ports()
{
    //https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
    check_TCP_port("Connection to Web (HTTP)", "80");
    check_TCP_port("Connection to Mail (SMTP) for routing between mail servers", "25");
    check_TCP_port("Connection to Mail (POP3)", "110");
    check_TCP_port("Connection to DNS", "53");
    check_TCP_port("Connection to (FTP) data transfer", "20");
    check_TCP_port("Connection to (FTP) data control", "21");

}
8 references
public void check_TCP_port(string name_port, string port)
{
    try
    {
        TcpClient tcp = new TcpClient();
        this.port = port;
        this.port_name = name_port;
        tcp.Connect(ip_address, Convert.ToInt16(port));
        port_state = "Online";

    }
    catch (Exception ex)
    {
        port_state = "Offline, " + ex.Message;
    }
}
```

Figure 4.18: TCP port scanning technique codes

```
public List<string[]> Check_MSFT_firewall()
{
    try
    {
        // Specify Windows Storage Management API namespace.
        ConnectionOptions options = new ConnectionOptions();
        options.Impersonation = System.Management.ImpersonationLevel.Impersonate;
        options.Username = user_name;
        options.Password = pass_word;
        var scope = new ManagementScope(@"\\" + ip_address + "\\root\\StandardCimv2", options);
        scope.Connect();

        var searcher = new ManagementObjectSearcher("SELECT * FROM MSFT_NetFirewallRule");
        searcher.Scope = scope;

        foreach (var filrwall_MSFT in searcher.Get())
        {
            //Our group used a tool to find out most of the elements in the server by wbemtest in windows + R
            string ElementName = filrwall_MSFT["ElementName"].ToString();
            string DisplayName = filrwall_MSFT["DisplayName"].ToString();
            string Status = filrwall_MSFT["Status"].ToString();

            string[] info_wmi = new string[] { ip_address, ElementName, DisplayName, Status };
            MSFT_firewall_list.Add(info_wmi); // store to List

        }
        return MSFT_firewall_list;
    }
    catch (Exception ex)
    {
        alert = ex.Message + " from " + "Check_MSFT_firewall";
        return MSFT_firewall_list;
    }
}
```

Figure 4.19: WMI Firewall codes

4.3.9 IIS and DNS monitoring

As shown in Figure 4.20, we can notice the integration between IIS and DNS services, as all information is displayed in the graph, like in PRTG. Besides, when the NOC team wants to verify the IP address of a specific website, they can use the DNS service to find out from the site itself.

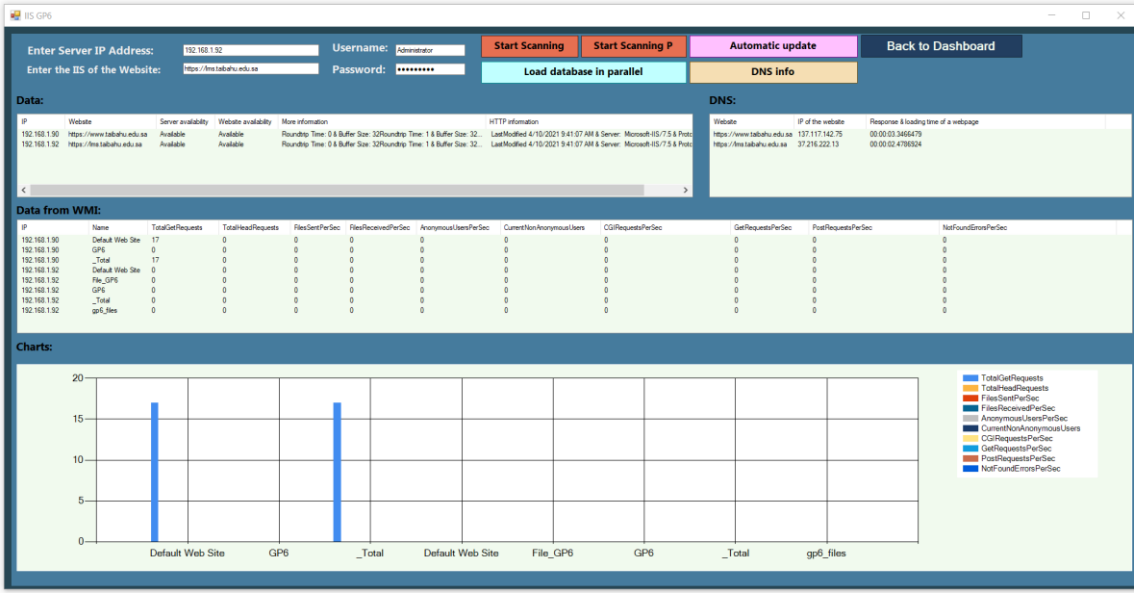


Figure 4.20: IIS and DNS monitoring

Figure 4.21 shows the method of how to get the information by using WMI and get these information to all the parameter in the web server. These parameters are using the parallel programming. After that, the Figure 4.23 shows how to get the IP address from the website. the reason that there are many possibilities for the inputs is to avoid the conflict with the last method. In Figure 4.22 shows the using of the HTTP with stopwatch.

```
public void WebService_WMI_Parallel()
{
    try
    {
        ConnectionOptions options = new ConnectionOptions();
        options.Impersonation = System.Management.ImpersonationLevel.Impersonate;
        options.Username = user_name;
        options.Password = pass_word;
        ManagementScope scope = new ManagementScope("\\\\" + ip_address + "\\root\\cimv2", options);
        scope.Connect();
        //Query system for web server information
        ObjectQuery query = new ObjectQuery("SELECT * FROM Win32_PerfFormattedData_W3SVC_WebService");
        ManagementObjectSearcher searcher = new ManagementObjectSearcher(scope, query);
        ManagementObjectCollection queryCollection = searcher.Get();
        List<ManagementObject> IIS_List = queryCollection.Cast<ManagementObject>().ToList();
        Parallel.ForEach(IIS_List, m =>
        {
            NameOfWebsite = m["Name"].ToString();
            TotalGetRequests = int.Parse(m["TotalGetRequests"].ToString());
            TotalHeadRequests = int.Parse(m["TotalHeadRequests"].ToString());
            FilesSentPerSec = int.Parse(m["FilesSentPerSec"].ToString());
            FilesReceivedPerSec = int.Parse(m["FilesReceivedPerSec"].ToString());
            AnonymousUsersPerSec = int.Parse(m["AnonymousUsersPerSec"].ToString());
            CurrentNonAnonymousUsers = int.Parse(m["CurrentNonAnonymousUsers"].ToString());
            CGIRequestsPerSec = int.Parse(m["CGIRequestsPerSec"].ToString());
            GetRequestsPerSec = int.Parse(m["GetRequestsPerSec"].ToString());
            PostRequestsPerSec = int.Parse(m["PostRequestsPerSec"].ToString());
            NotFoundErrorsPerSec = int.Parse(m["NotFoundErrorsPerSec"].ToString());
        });//eachfor P
          }//try
    catch (Exception ex)
    {
        alert = ex.Message + " from " + "IIS_W3SVC_WebService_WMI_Parallel";
    }//catch
}//WebService_WMI_Parallel
```

Figure 4.21: WMI IIS codes

```
public string HTTP_info()
{
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(website);
    try
    {
        Stopwatch timer = new Stopwatch();
        timer.Start();

        HttpWebResponse response = (HttpWebResponse)request.GetResponse();

        timer.Stop();

        TimeSpan timeTaken = timer.Elapsed;

        return timeTaken.ToString();

    }
    catch (Exception ex)//add by Mohammad
    {
        alert = ex.Message + " From HTTP_info";
        return ex.Message;
    }
}
```

Figure 4.22: HTTP codes

```
3 references
public string DNS_info()
{
    string website_new = website;
    if (website.Contains("https://www."))
    {
        website_new = website.Remove(website.IndexOf("https://"), "https://www.".Length);
    }

    else if (website.Contains("http://www."))
    {
        website_new = website.Remove(website.IndexOf("http://"), "http://www.".Length);
    }

    if (website.Contains("https://"))
    {
        website_new = website.Remove(website.IndexOf("https://"), "https://".Length);

    }

    else if (website.Contains("http://"))
    {
        website_new = website.Remove(website.IndexOf("http://"), "http://".Length);
    }

    if (website_new.Contains("/"))
    {
        website_new = website_new.Remove(website_new.IndexOf("/"), "/".Length);
    }
    try
    {
        var address = Dns.GetHostAddresses(website_new)[0]; //old
        return address.ToString();
    }
    catch (Exception ex)
    {
        alert = ex.Message + " From DNS_info";
        return ex.Message;
    }

}
3 references
```

Figure 4.23: DNS codes

4.3.10 OS monitoring

The Figure 4.24 shows the result of OS monitoring. We notice the different resources are presented in the chart like CPU, RAM, Disk Usage, and Network.

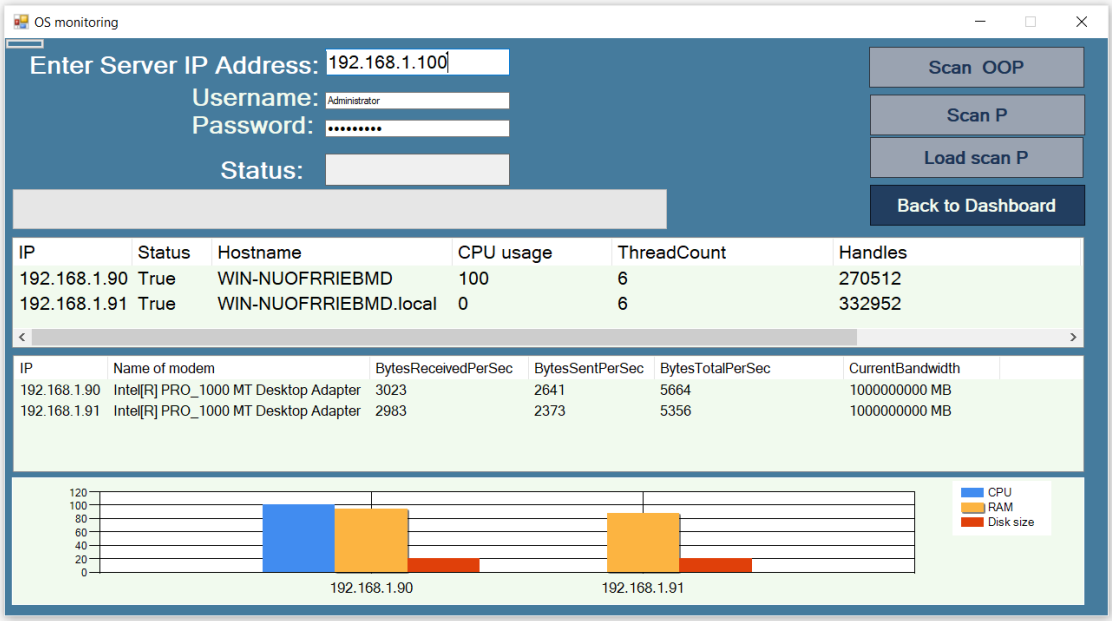


Figure 4.24: OS monitoring window

The Figure 4.25 shows that the codes are using WMI for most of the methods. Also, we used SNMP to differentiate the system between Windows or Linux as in Figure 4.26.

Appendix C shows the steps to enable SNMP and WMI for Windows system. In addition, appendix D shows the tool that can help to display the results of a query to get CPU utilization.

```
public void get_CPU_Parallel()
{
    try
    {
        ConnectionOptions options = new ConnectionOptions();
        options.Impersonation = System.Management.ImpersonationLevel.Impersonate;
        options.Username = user_name;
        options.Password = pass_word;
        ManagementScope scope = new ManagementScope("\\\\" + ip_address + "\\root\\cimv2", options);
        scope.Connect();
        ObjectQuery wmicpus = new WqlObjectQuery("SELECT * FROM Win32_PerfFormattedData_PerfOS_Processor Where Name = '_Total'");
        ManagementObjectSearcher cpus = new ManagementObjectSearcher(scope, wmicpus);
        ManagementObjectCollection cpu_queryCollection = cpus.Get();
        List<ManagementObject> cpu_List = cpu_queryCollection.Cast<ManagementObject>().ToList();
        Parallel.ForEach(cpu_List, (cpu, state) =>
        {
            uint PercentProcessorTime = Convert.ToUInt32(cpu["PercentProcessorTime"]);
            Cpu_usage = PercentProcessorTime.ToString();
            state.Stop();
        });
    }//try
    catch (Exception ex)
    {
        alert = ex.Message + " from " + "get_CPU_Parallel";
    }//catch
}//get_CPU_Parallel
```

Figure 4.25: WMI CPU code

```
public string Detect_SW_SNMP(string ip_address)
{
    string sysDescr = "n/a";
    OctetString community = new OctetString("public");
    AgentParameters param = new AgentParameters(community);
    param.Version = SnmpVersion.Ver2;
    IpAddress agent = new IpAddress(ip_address);
    UdpTarget target = new UdpTarget((IPAddress)agent, 161, 2000, 1);
    Pdu pdu = new Pdu(PduType.Get);
    pdu.VbList.Add("1.3.6.1.2.1.1.1.0"); //sysDescr
    pdu.VbList.Add("1.3.6.1.2.1.1.2.0"); //sysObjectID
    pdu.VbList.Add("1.3.6.1.2.1.1.3.0"); //sysUpTime
    SnmpV2Packet result = (SnmpV2Packet)target.Request(pdu, param);
    if (Monitoring_GP6_M_A_2021.PingHost(ip_address))
    {
        Console.WriteLine("Online by SNMP to: " + ip_address);
    }
    else
    {
        Console.WriteLine("ERROR: You have Some TIMEOUT issue");
    }
    if (result != null)
    {
        if (result.Pdu.ErrorStatus != 0)
        {
            // agent reported an error with the request
            Console.WriteLine("Error in SNMP reply. Error {0} index {1}",
                result.Pdu.ErrorStatus,
                result.Pdu.ErrorIndex);
        }
        else
        {
            sysDescr = result.Pdu.VbList[0].Oid.ToString() +
                SnmpConstants.GetTypeName(result.Pdu.VbList[0].Value.Type) +
                result.Pdu.VbList[0].Value.ToString();
            label_uptime.Text =  "UpTime: " + result.Pdu.VbList[2].Value.ToString();
        }
    }
    else
    {
        Console.WriteLine("No response received from SNMP agent.");
        sysDescr = "No response received from SNMP agent.";
    }
    return sysDescr;
}//Method
```

Figure 4.26: SNMP OS code

4.3.11 Eventlog monitoring

Figure 4.27 shows the results of Eventlog monitoring. Time is important for this service, so a calendar has been added to make it easier for the network administrator to get the information by choosing a specific time. Also, after reading the information, you can get to display a pie chart for the details of the events.

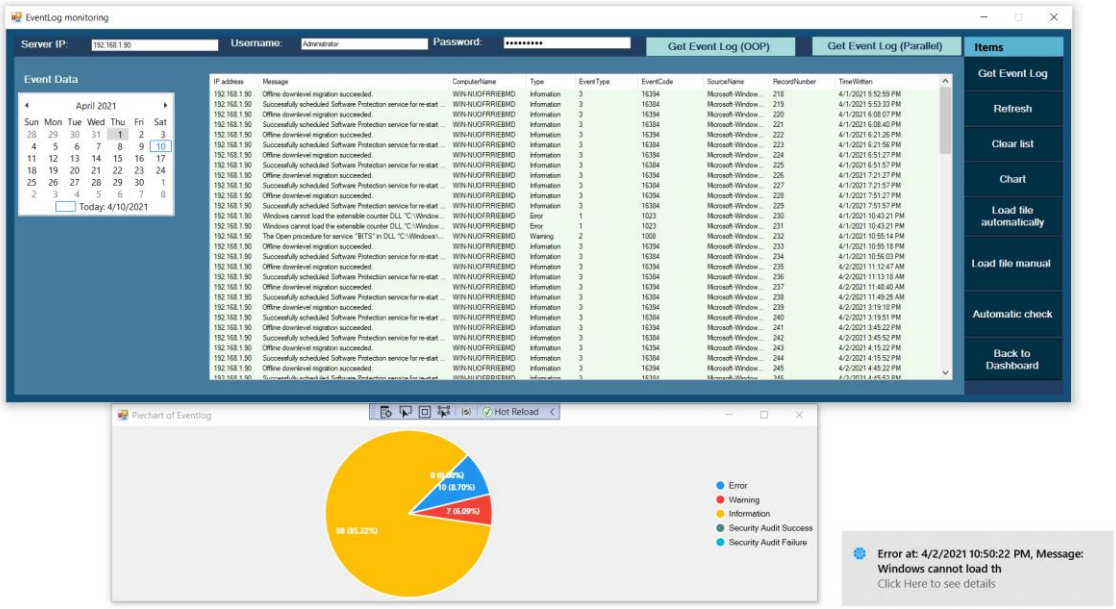


Figure 4.27: Eventlog and Pie chart window

The codes in Figure 4.28 are using WMI like some previous services. And it implements a filter to collect a specific event by comparing the event types.

```
public void scan_event_log_Parallel(string Events_selected_for_viewing, bool automatic_scan, string monthCalendar1)
{
    try
    {
        if (base.PingHost()) // base in C# , super in java
        {
            var conOpt = new ConnectionOptions();
            conOpt.Impersonation = ImpersonationLevel.Impersonate;
            conOpt.EnablePrivileges = true;
            conOpt.Username = user_name;
            conOpt.Password = pass_word;
            var scope = new ManagementScope(String.Format(@"\\{0}\ROOT\CIMV2", ip_address), conOpt);
            scope.Connect();
            bool isConnected = scope.IsConnected;
            if (isConnected)
            {
                string dateTime = getDmtfFromDateTime(DateTime.Today.Subtract(new TimeSpan(1, 0, 0, 0)));  /* entire day */
                //  best practice :) ; DRY (Don't repeat yourself)
                if (automatic_scan)
                {
                    dateTime = getDmtfFromDateTime(DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss tt"));
                }
                else
                {
                    dateTime = getDmtfFromDateTime(monthCalendar1); // DateTime specific
                }
                SelectQuery query = new SelectQuery("Select * from Win32_NTLogEvent Where Logfile = 'Application' and TimeGenerated >='" + dateTime + "'");
                ManagementObjectSearcher searcher = new ManagementObjectSearcher(scope, query);
                ManagementObjectCollection logs = searcher.Get();
                List<ManagementObject> logsList = logs.Cast<ManagementObject>().ToList();
                Parallel.ForEach(logsList, log =>
                {
                    Console.WriteLine($"thread = {Thread.CurrentThread.ManagedThreadId}");
                if (string.Equals(Events_selected_for_viewing, "All"))
                    {
                    string[] Event_log = new string[] { ip_address, log["Message"].ToString(), log["ComputerName"].ToString(),
                     log["Type"].ToString(), log["EventType"].ToString(), log["EventCode"].ToString(),
                     log["SourceName"].ToString(), log["RecordNumber"].ToString()
                     , getDateTimeFromDmtfDate(log["TimeWritten"].ToString()) };
                        Eventlog_list.Add(Event_log); // store to List
                }
                //filters
                else if (string.Compare(Events_selected_for_viewing, log["EventType"].ToString()) == 0)
                    {
                    string[] Event_log = new string[] { ip_address, log["Message"].ToString(), log["ComputerName"].ToString(),
                     log["Type"].ToString(), log["EventType"].ToString(), log["EventCode"].ToString(),
                     log["SourceName"].ToString(), log["RecordNumber"].ToString()
                     , getDateTimeFromDmtfDate(log["TimeWritten"].ToString()) };

                        Eventlog_list.Add(Event_log); // store to List
                }
            });


            }
            //To close the connection to become like Syslog | Ahmed & Mohammad in meet 18 - 2 - 2021
            scope.Path = new ManagementPath(string.Empty);
        }//if ping
        else
```

Figure 4.28: WMI and filter codes

4.3.12 Discover monitoring

The role of the discovery stage is when using our program enters a network with a new topology. This window, as in Figure 4.29 helps to explore based on the extent of the subnet, and the availability and device name appear so that the network administrator can know the existing servers.

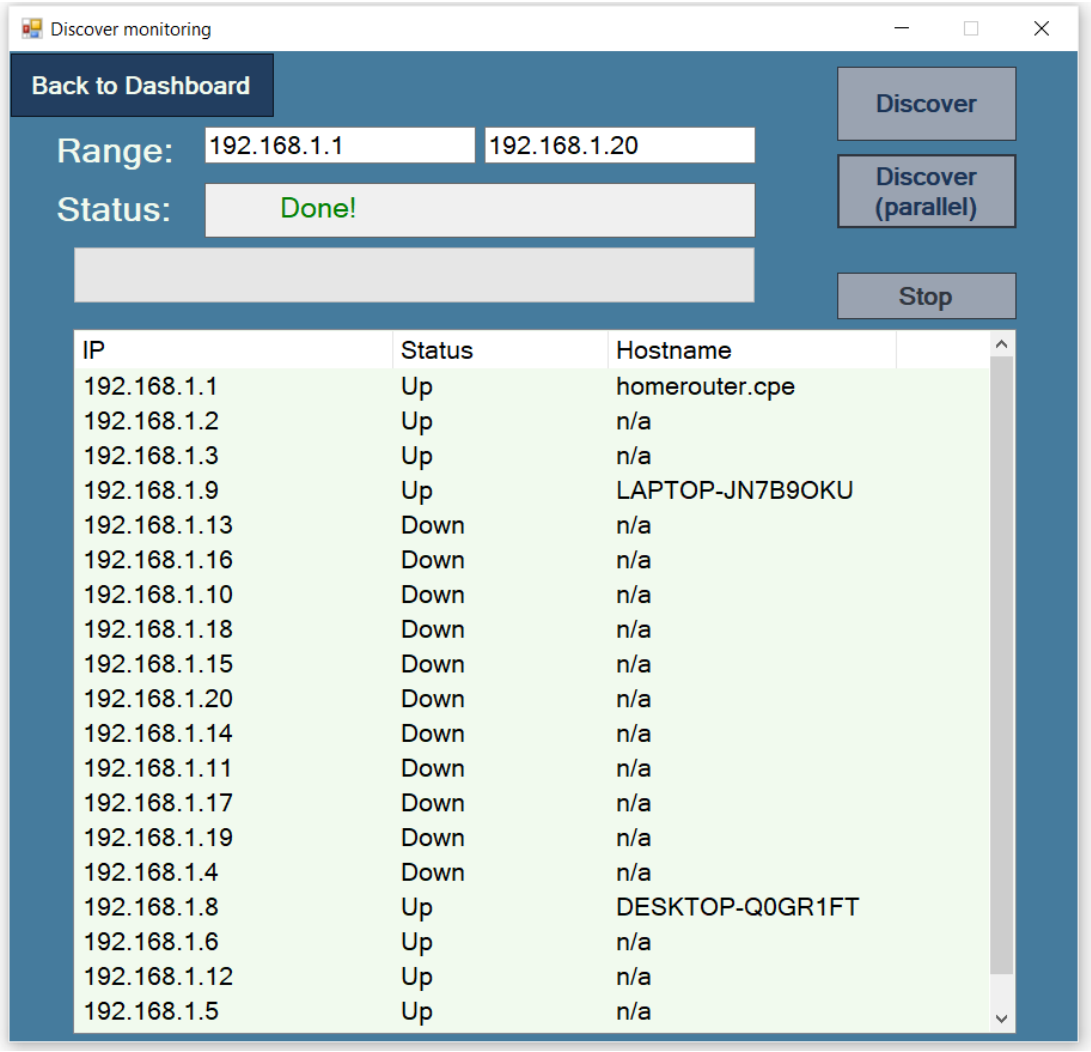


Figure 4.29: Discover window

In Figure 4.30 the code depends on ping inspection, which in turn depends on the ICMP protocol, with the use of programming techniques in parallel to inspect a group of servers at the same time, as we mentioned in Table 3.2.

```
public void Parallel_scan_discover_GP6(string start, string end)
{
    try
    {
        //Split IP string into a 4 part array
        string[] startIPString = start.Split('.');
        int[] startIP = Array.ConvertAll<string, int>(startIPString, int.Parse); //Change string array to int array
        string[] endIPString = end.Split('.');
        int[] endIP = Array.ConvertAll<string, int>(endIPString, int.Parse);
        int count = 0; //Count the number of successful pings
        IPAddress addr;
        IPHostEntry host;
        //Progress bar
        progressBar1.Maximum = 254;
        progressBar1.Value = 0;
        Discover_list.Items.Clear();
        //Parallel
        int i = start[2];
        int j = endIP[2] - 1;
        int y = startIP[3];
        //CancellationTokenSource cts = new CancellationTokenSource();

        Parallel.For(startIP[3], endIP[3] + 1, count_p =>
        {
            //For debug
            Console.WriteLine($"value of count = {count_p}, thread = {System.Threading.Thread.CurrentThread.ManagedThreadId}");
            string ipAddress = startIP[0] + "." + startIP[1] + "." + startIP[2] + "." + count_p; //Convert IP array back into a string
            string endIPAddress = endIP[0] + "." + endIP[1] + "." + endIP[2] + "." + (endIP[3] + 1); // +1 is so that the scanning stops at the correct range
            lblStatus.ForeColor = System.Drawing.Color.Green; //Set status label for current IP address
            lblStatus.Text = "Scanning: " + ipAddress;
            //Log pinged IP address in listview grabs DNS information to obtain system info
            if (PingHost(ipAddress))
            {
                try
                {
                    addr = IPAddress.Parse(ipAddress);
                    host = Dns.GetHostEntry(addr);
                    Discover_list.Items.Add(new ListViewItem(new String[] { ipAddress, "Up", host.HostName })); //Log successful pings
                    count++;
                }
                catch
                {
                    Discover_list.Items.Add(new ListViewItem(new String[] { ipAddress, "Up", "n/a" })); //Log successful pings
                    count++;
                }
            }
            else
            {
                Discover_list.Items.Add(new ListViewItem(new String[] { ipAddress, "Down", "n/a" })); //Log unsuccessful pings
            }
            System.Threading.Thread.Sleep(10);//Sleep the loop for 10 miliseconds
        });
```

Figure 4.30: Discover codes

4.3.13 Database management

Controlling the database data that contains connection and authentication details for each server in the network, can add a server to it or take an action on a specific server like send a reboot signal through the window as in Figure 4.31.

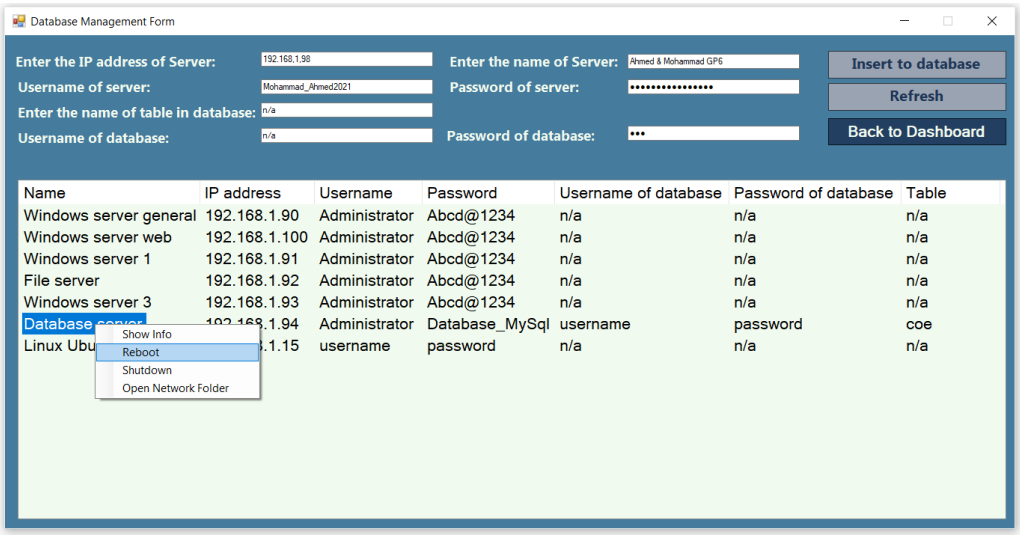


Figure 4.31: Database management window

4.3.14 Polling with Connectivity

The principle used in our tool is to check the state of our servers and services, and this principle is based on the method of polling.

The idea of our tool is to show alerts in stages, the first stage is to inform the network administrator on his device, after that the second stage, which will send an email about the problem to the network administrator and the NOC team. If there is no response from them; the tool will have to take an action such as restarting the server. In section (A) in the Figure 4.32, the notification appears on the network administrator's device, but in Figure 4.33 we see the email for the NOC team.

In section (B) in the Figure 4.32 shows that if the network administrator or the NOC team wishes to take responsibility for fixing the network problems; We have provided an optional mode in our tool that they can activate to solve the problem. This method requires restarting the server if it encounters problems that have not been solved after a period of time, as shown in Table 3.11.

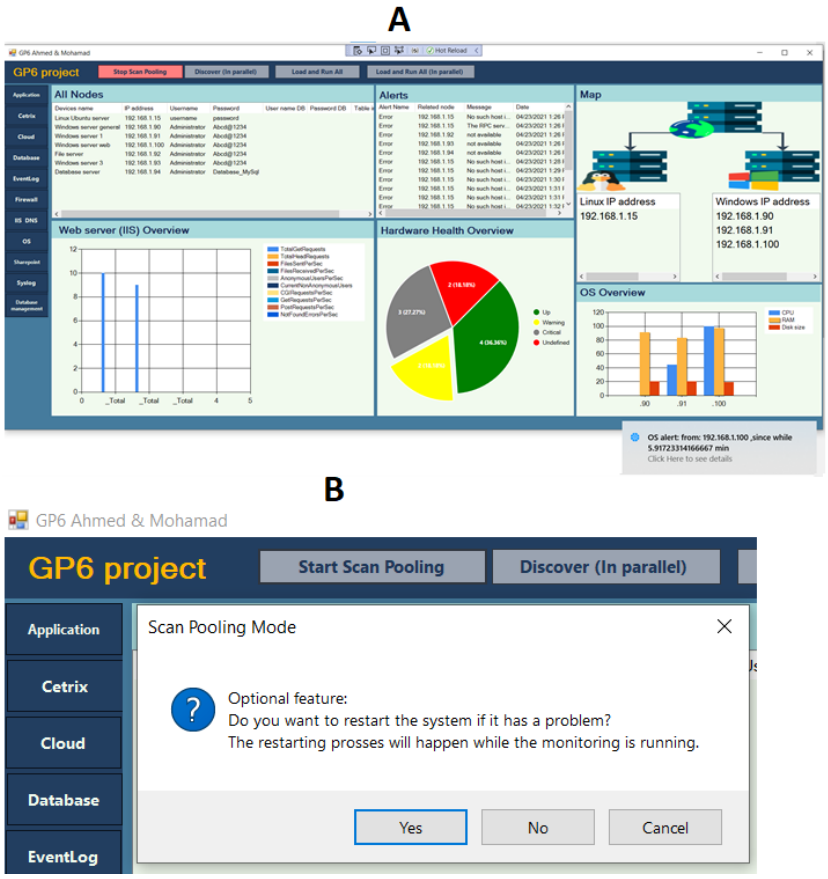


Figure 4.32: Alert on polling

In Figure 4.33, we can see the details of the message that was sent for reasons such as increased CPU and RAM consumption. We have designed an interface for the message using the following web languages Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) to send a clear message that contains the details of the problem and in which server it is located in. In addition to this, it gives us the server details according to the defect in it. Furthermore, what distinguishes our tool is that it shows the problem and at which layer of the network it is.

The site that helped us in designing the email message is [24], after that we modified the files in a way that suits us to make them dynamic according to the problem.

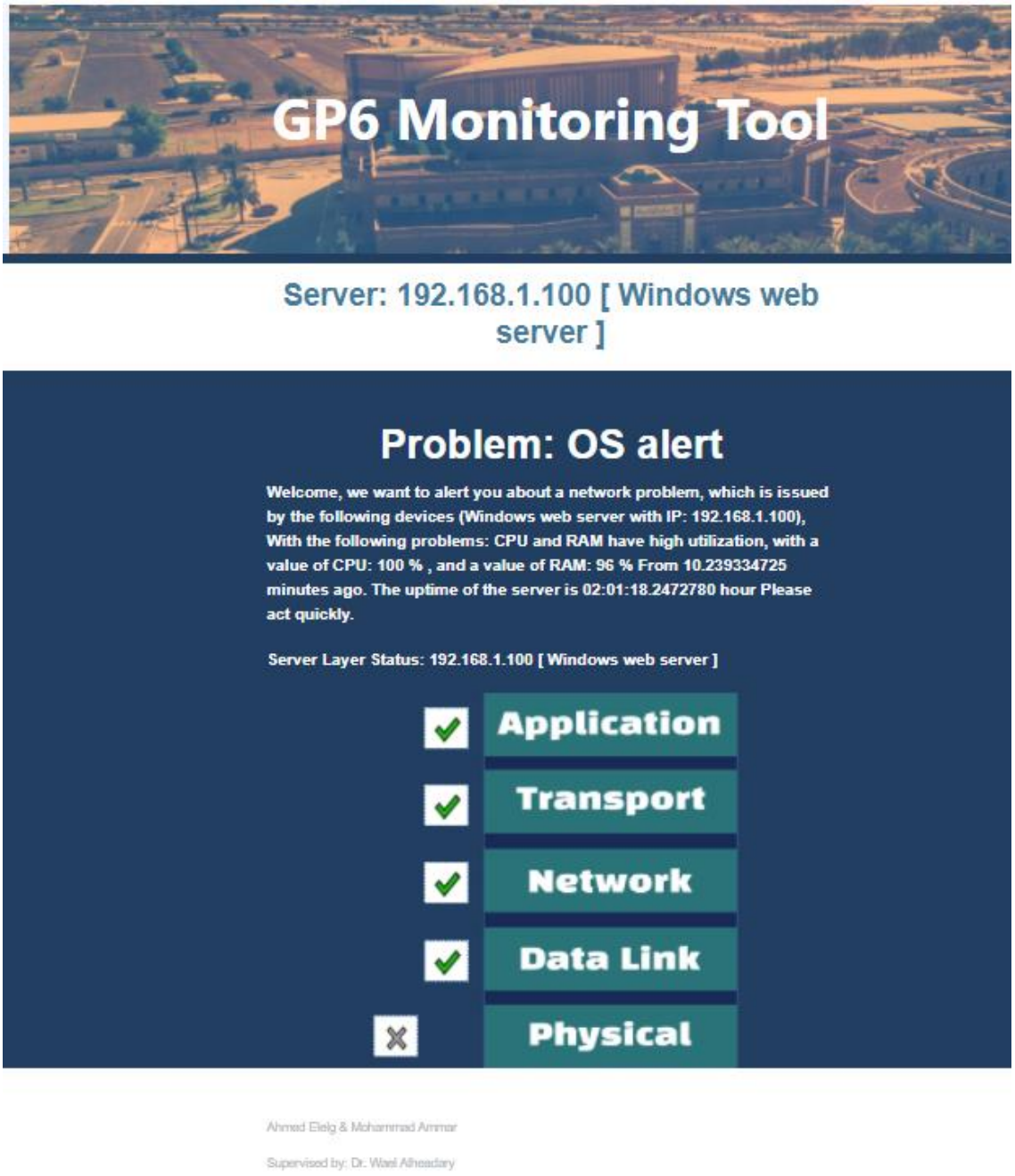


Figure 4.33: Email on polling

Figure 4.34 shows the design of the message and how it appears on the browser and the mobile phone with several experiences with email for several additional services linked to layers, for example in Figure 4.35, with the Transport layer when trying to connect via UDP via SNMP. In addition, in Figure 4.36 to the Application layer with web server or database problems. Also, in Figure 4.37 when there are problems with the connection in the Network layer, we will know that by getting the length of the output packet queue (in packets). If this packet has a value longer than 2, delays are being experienced and the bottleneck should be found and eliminated [25].

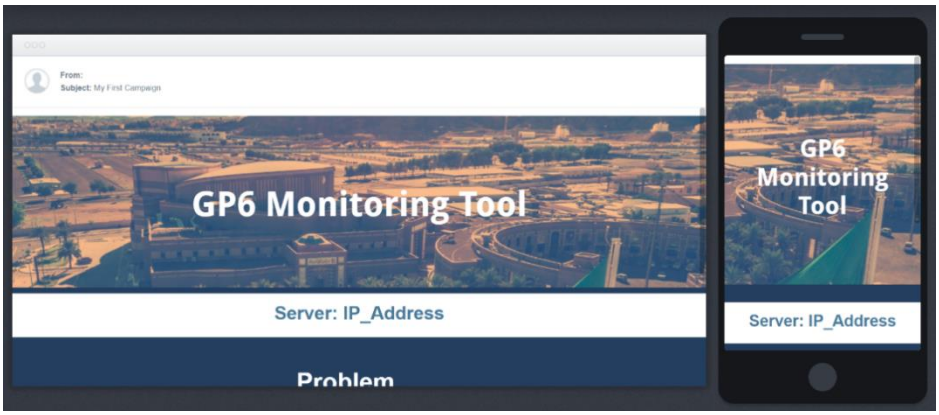


Figure 4.34: Design of the email message

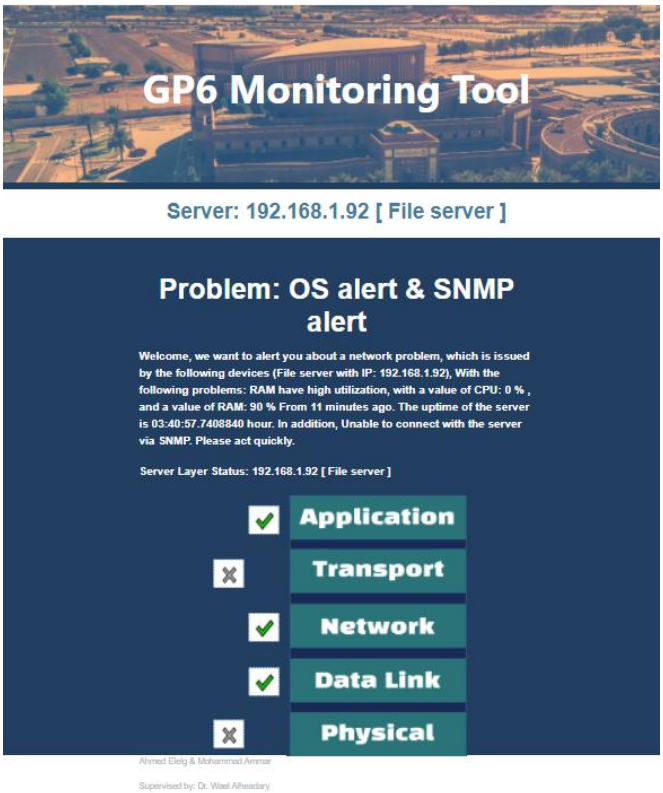


Figure 4.35: Error detected in Transport layer

Figure 4.36: Error detected in Application layer

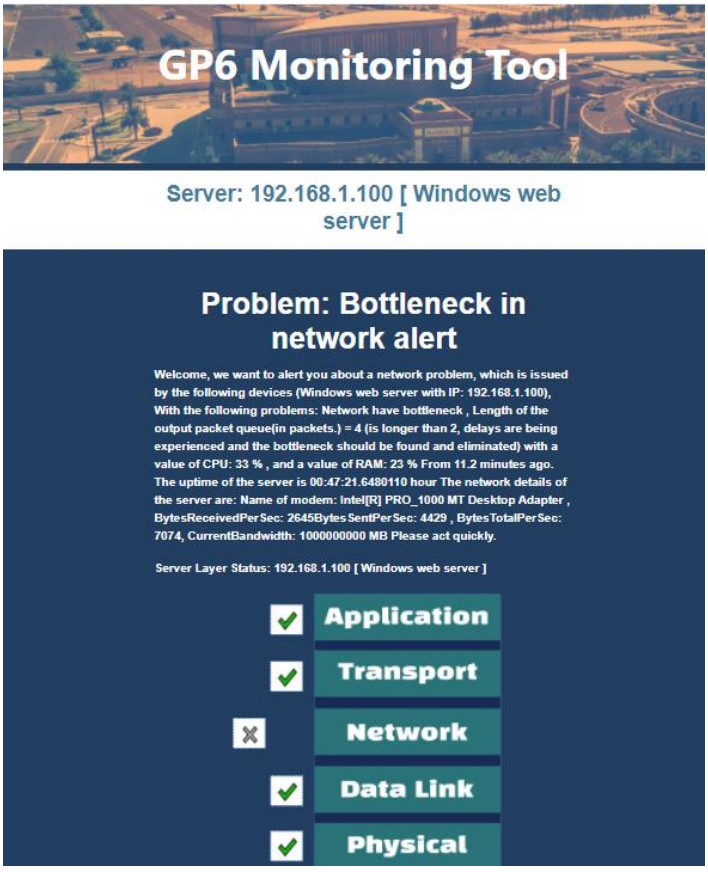


Figure 4.37: Error detected in Network layer

In the end, Figure 4.38 shows an email for the status of the password recovery request. Moreover, this email will be sent to the help disk team to deal with the request, and they will communicate with the user.

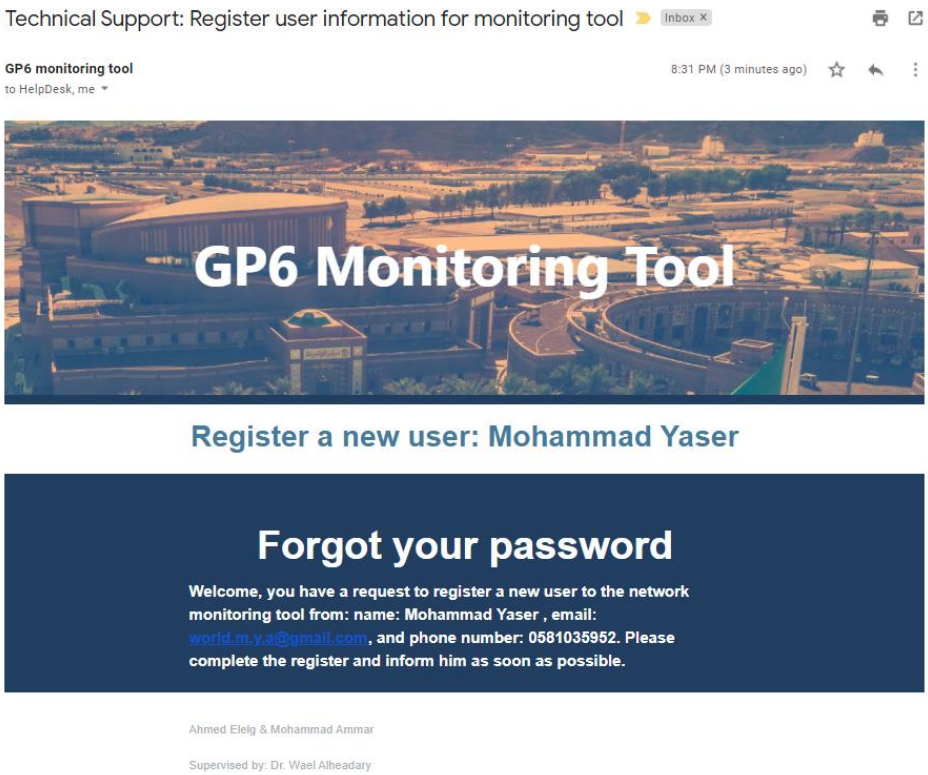


Figure 4.38: An example of Forgot your password email

The service priorities in reporting and notifications in our network monitoring tool depend on that all the services have all the priority because they deal with the whole of the University.

4.3.15 Comparison with PRTG

We wanted to compare our results with PRTG, as we added a virtual server to it so that we can compare the results between the two tools. In Figure 4.39 we see the results that are similar to our monitoring tool in Figure 4.40, but we notice the different units of the results. Hence, there is an interface in our program that shows the Network layer.

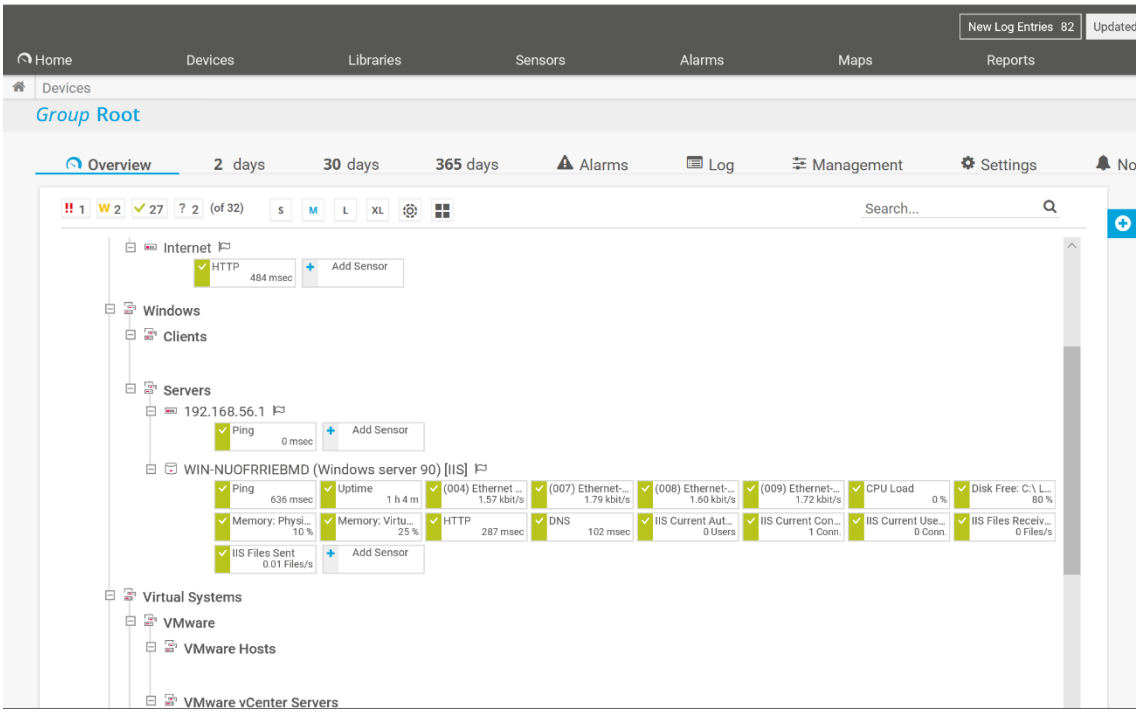


Figure 4.39: Overview of PRTG to server devices

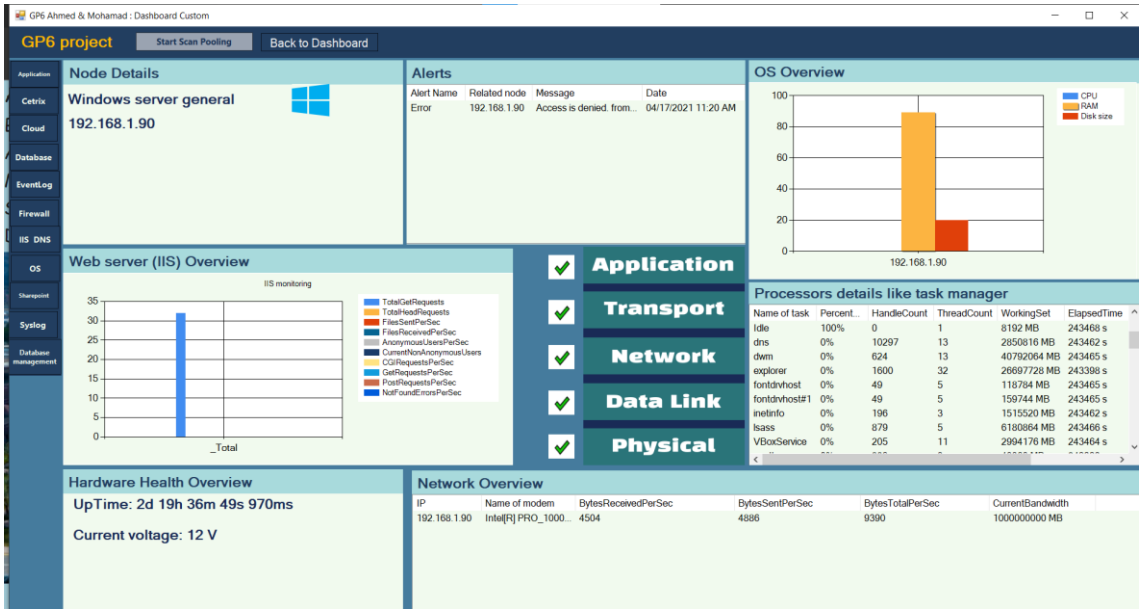


Figure 4.40: Our software tool with a server result

In the second experiment, the simulation process was carried out on the server using the AIDA64 program, which helps to increase the consumption of the processor and memory. And we can see the results in our program as shown in Figure 4.41, compared to PRTG in Figure 4.42. We have made sure to increase the reliability of the experiment at the same time as shown in the taskbar for two different devices.

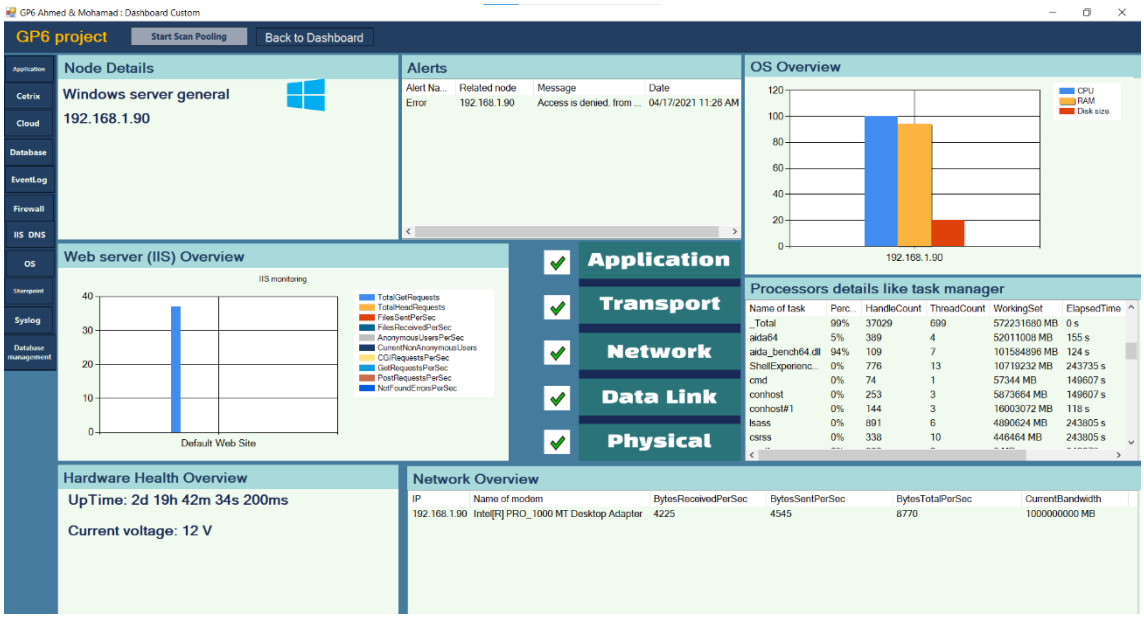


Figure 4.41: Our software tool with server results after simulation

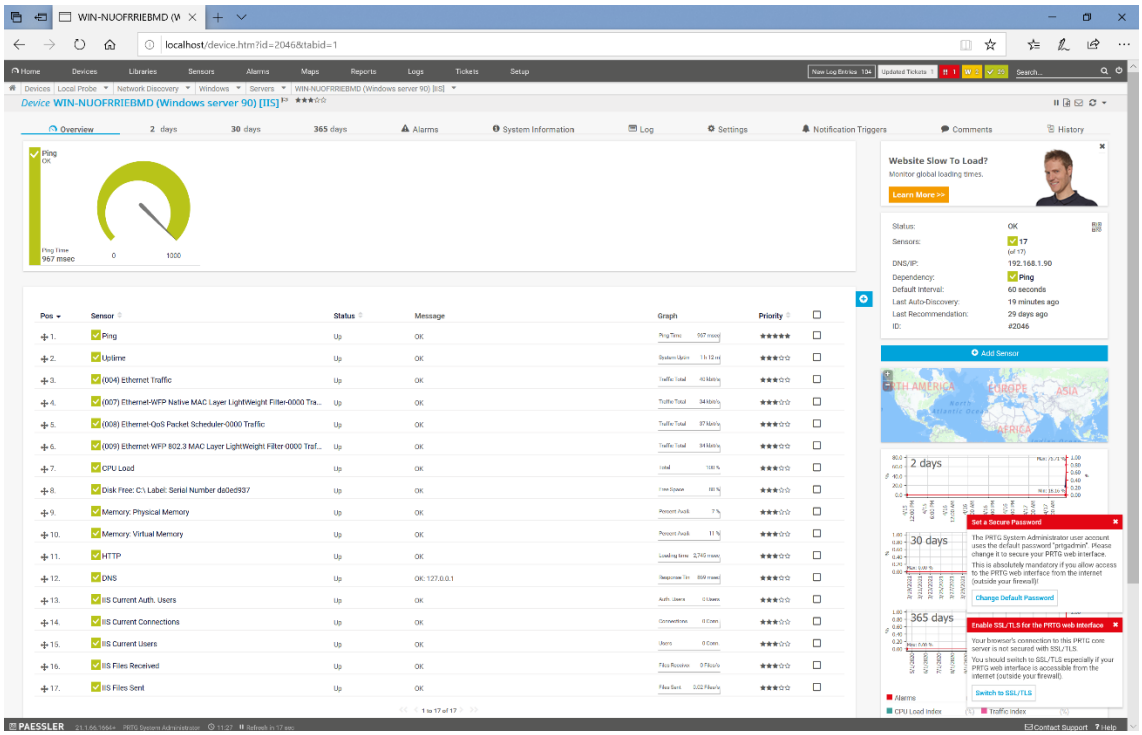


Figure 4.42: Details from PRTG to a server device after simulation

Our monitoring tool is characterized by the presence of features that are not present in the competing paid services, such as identifying the problem and in which layer it is located, monitoring modes, and sending a signal to restart the server.

4.4 Real infrastructure implementation

With the help of the Deanship of Information Technology through Eng. Maher Ezzi, who allowed us to try some services on the real university network so that we could prove the program's efficiency.

4.4.1 Cloud monitoring

In Figure 4.43, this was an experiment hold on a real infrastructure. The cloud service is running on Microsoft Azure, and our tool was able to get the key information for monitoring by using a method that is similar to the one we implemented for the databases.

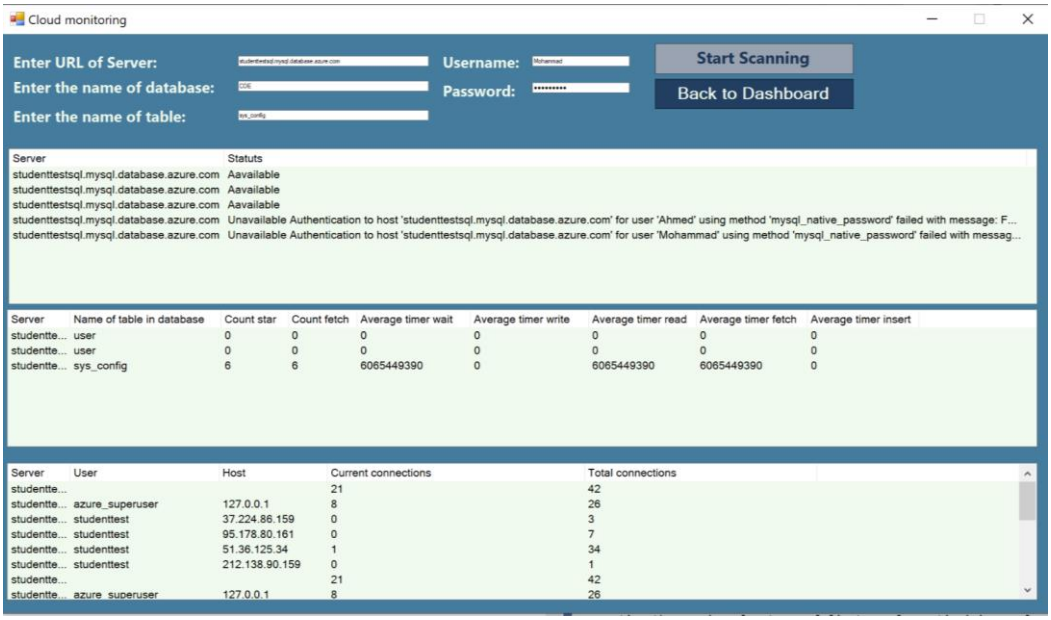


Figure 4.43: Cloud monitoring result in real infrastructure

The codes in figure 4.44, is start with connecting to the university's azure server and then the databases to read the basic information and to get an impression on the state of the traffic. Hence, we notice that there are some devices that consume more than others. It may be possible to track if a student communicates with a malicious software method to compress the server.

```
public void read_Azure_performance_accounts()
{
    try
    {
        var builder = new MySqlConnectionStringBuilder
        {
            Server = ip_address,
            Port = 3306,
            Database = "performance_schema",
            UserID = User_name_cloud,
            Password = Pass_word_cloud,
            SslMode = MySqlSslMode.Preferred,
        };
        using (var conn = new MySqlConnection(builder.ConnectionString))
        {
            Console.WriteLine("Opening connection");
            conn.Open();
            using (var command = conn.CreateCommand())
            {
                Connection = "Aavailable ";
                string query = "SELECT* FROM `accounts`";
                command.CommandText = query;
                using (var reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        string USER = reader["USER"].ToString();
                        string HOST = reader["HOST"].ToString();
                        string CURRENT_CONNECTIONS = reader["CURRENT_CONNECTIONS"].ToString();
                        string TOTAL_CONNECTIONS = reader["TOTAL_CONNECTIONS"].ToString();
                        string[] users_info = new string[] { ip_address, USER, HOST, CURRENT_CONNECTIONS, TOTAL_CONNECTIONS };
                        Users_Azure_list.Add(users_info);// store to List
                    }
                }
            }
            Console.WriteLine("Closing connection, read_Azure_performance_accounts");//foe debug
        }
    }//try
    catch (Exception ex)
    {
        Connection = "Unavailable " + ex.Message;
        alert = ex.Message + " From read_Azure_performance_accounts";
    }
}//read_Azure_performance_accounts
```

Figure 4.44: Cloud monitoring codes

4.4.2 SharePoint and Citrix monitoring

The implementation was supervised by the Deanship of Information Technology at Taibah University, but we were unable to visualize the results because it is sensitive data. Additionally, there was an increase in the continuous use of resources on the Citrix service towards the end of the semester due to the use of programs such as MATLAB and Microsoft projects across Citrix.

As for SharePoint, there was a challenge with the additional login provided by the university, which is of the type of ADFS.

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The network monitoring system is mandatory for any organization. Additionally, it has countless benefits to maintain the efficiency and the stability of the network performance.

The main point of this project is to observe and monitor application network services. Moreover, our proposed tool assists any network administrator to analyse and evaluate the behaviour of all offered services. Furthermore, it always helps the application developer to rectify the bugs for all the utilized network services.

We must improve the position of our university by preserving the sustainability of its network services. To sum up, in a short period of time, we have been able to build a tool that is capable of monitoring and interacting with the network and has the ability to run with parallel programming. It also contains graphical charts that help to describe the state of the server. Furthermore, we manage to merge with different programming languages like HTML and CSS with C#.

Finally, the existing services in the monitoring tool are free of charge and are subject to development and updating over time.

5.2 Future Work

We hope in the future to add AI to the monitoring system, which aims to analyse data, make decisions and act automatically to maintain the stability of the network and applications all the time. In addition to that, implement our program on a web server.

REFERENCES

[1] A. K. Rajput, R. Tewani and A. Dubey, “The helping protocol “DHCP”,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2016.

[2] G. S. Hura and M. Singhal, Data and Computer Communications: Networking and Internetworking, Boca Raton, FL: CRC Press, 2001, p. 262.

[3] "Network Management System: Best Practices White Paper," Cisco, 10 8 2018. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/availability/high-availability/15114-NMS-bestpractice.html. [Accessed 1 12 2020].

[4] J. F. Kurose and K. . W. Ross, Computer Networking: A Top-Down Approach (6th Edition), vol. 6, Addison-Wesley Professional, 2012.

[5] A. Kijazi and M. Kisangiri, "A Step on Developing Network Monitoring Tools," 1 9 2014.

[6] B. Hale, "Network management – back to the basics," [Online]. Available: http://content.solarwinds.com/creative/pdf/Whitepapers/Network_Management_-_Back_to_the_Basics.pdf. [Accessed 12 11 2020].

[7] S. Saha, "OBSAI Interoperability in Multi-Vendor WiMAX Base Station Architecture Environment," 19 6 2009. [Online]. Available: https://people.kth.se/~maguire/DEGREE-PROJECT-REPORTS/090703-Sumanta-Saha-with-cover.pdf. [Accessed 10 12 2020].

[8] D. Mauro and K. Schmidt, Essential SNMP: Help for system and network administrators, vol. 2, O'Reilly Media, 2005.

[9] M. M. Lavy and A. J. Meggitt, Windows Management Instrumentation (WMI), Indianapolis, IN: Sams Publishing, 2001, p. 3.

[10] P. Swathi and M. Ahmed, “Agent based discovery and system monitoring,” in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2017.

[11] L. Kalita, “Socket Programming,” *International Journal of Computer Science and Information Technologies (IJCSIT),* p. 4804, 2014.

[12] V. Prenosil, I. Ghafir and J. Svoboda, "Network Monitoring Approaches: An Overview," *International Journal of Advances in Computer Networks and Its Security– IJCNS,* pp. 88-93, 10 2015.

[13] R. Khan, S. U. Khan, R. Zaheer and M. I. Babar, "An Efficient Network Monitoring and Management System," *International Journal of Information and Electronics Engineering,* 1 2013.

[14] R. Johnson and N. E. Elizabeth, "Network's server monitoring and analysis using Nagios," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2017.

[15] A. Kore, M. Bha, O. Gorana, A. Ghugul and S. Saha, “An Efficient Network Monitoring and Management System,” *Int. j. inf. electron. eng.,* 2019.

[16] V. Formoso, F. Cacheda, V. Carneiro and J. Valiño, "Open Source Tool For Network Monitoring," *IJBDCN,* pp. 22-39, 1 2009.

[17] Paessler, "PRTG manual," 2020. [Online]. Available: https://manuals.paessler.com/prtgmanual.pdf. [Accessed 11 12 2020].

[18] D. Manvar, M. Mishra and A. Sahoo, "Low cost computing using virtualization for Remote Desktop," in *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, Bangalore, India, 2012.

[19] Microsoft, “Monitoring and maintaining SharePoint Server 2013,” 25 8 2017. [Online]. Available: https://docs.microsoft.com/en-us/sharepoint/administration/monitoring-and-maintaining. [Accessed 22 11 2020].

[20] “Syslog analyzer PRTG - Log messages at a glance!,” Paessler, [Online]. Available: https://www.paessler.com/syslog-monitoring. [Accessed 2 12 2020].

[21] Information Resources Management Association (IRMA), Cloud technology: Concepts, methodologies, tools, and applications, Hershey, PA: Information Science Reference, 2014, p. 406.

[22] R. L. Bocchino Jr, V. S. Adve, S. V. Adve and M. Snir, "Parallel programming must be deterministic by default," [Online]. Available: https://www.usenix.org/legacy/events/hotpar09/tech/full_papers/bocchino/bocchino.pdf. [Accessed 2 4 201].

[23] R. Blum, C# Network Programming, Indianapolis, IN: Sybex, 2006, p. 153.

[24] "campaignmonitor," CM Group, [Online]. Available: https://www.campaignmonitor.com/email-templates/. [Accessed 22 4 2021].

[25] Microsoft, "Win32_PerfFormattedData_Tcpip_NetworkInterface class," [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/aa394293(v=vs.85). [Accessed 24 4 2021].

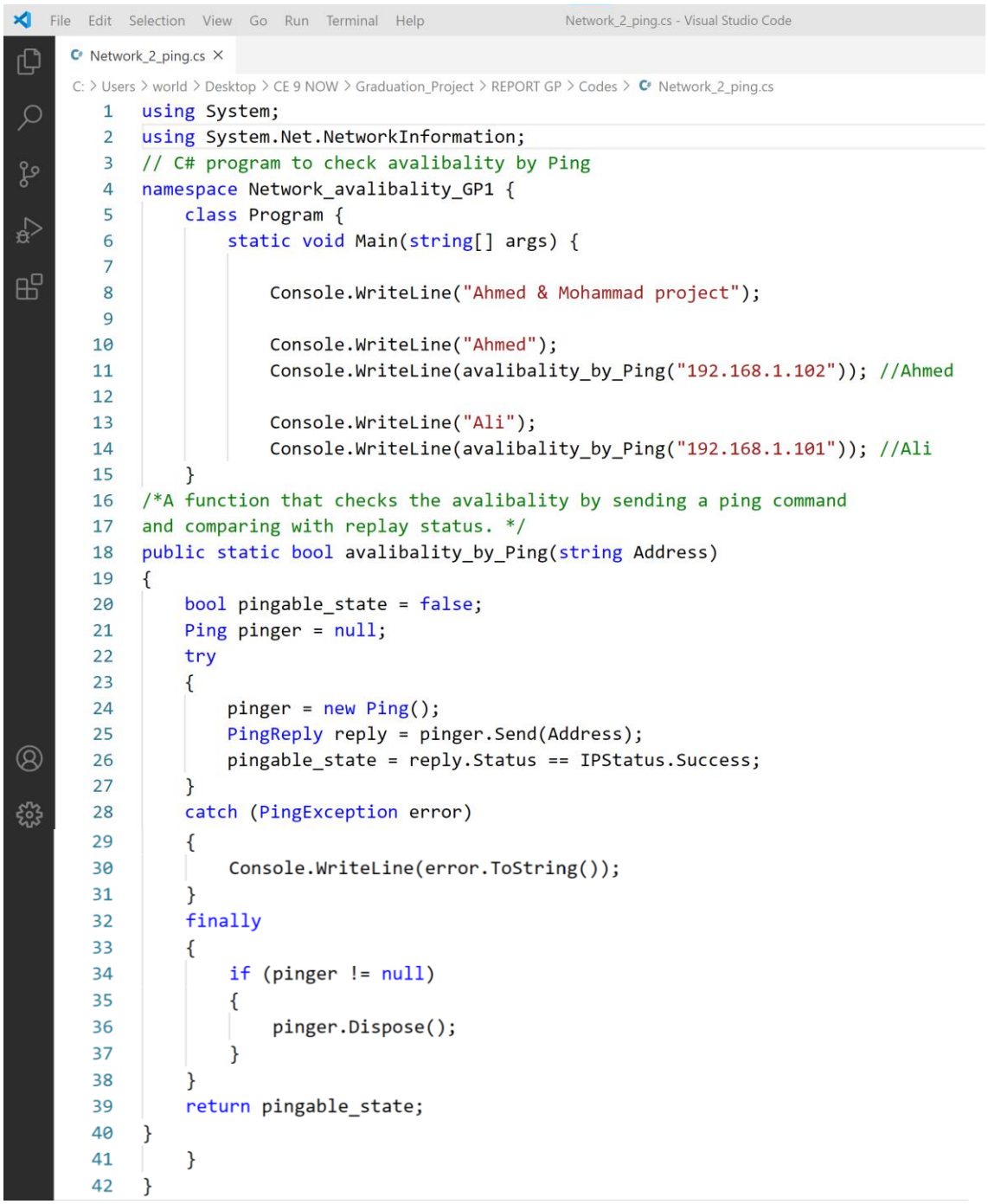
[26] J. Darlington, A. J. Field, P. G. Harrison, P. H. J. Kelly, D. W. N. Sharp, Q. Wu and R. L. While, Parallel programming using skeleton functions, vol. 694, Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 146-160.

[27] Cloudflare, "What Is DNS? | How DNS Works," [Online]. Available: https://www.cloudflare.com/learning/dns/what-is-dns/. [Accessed 29 3 2021].

APPENDICES

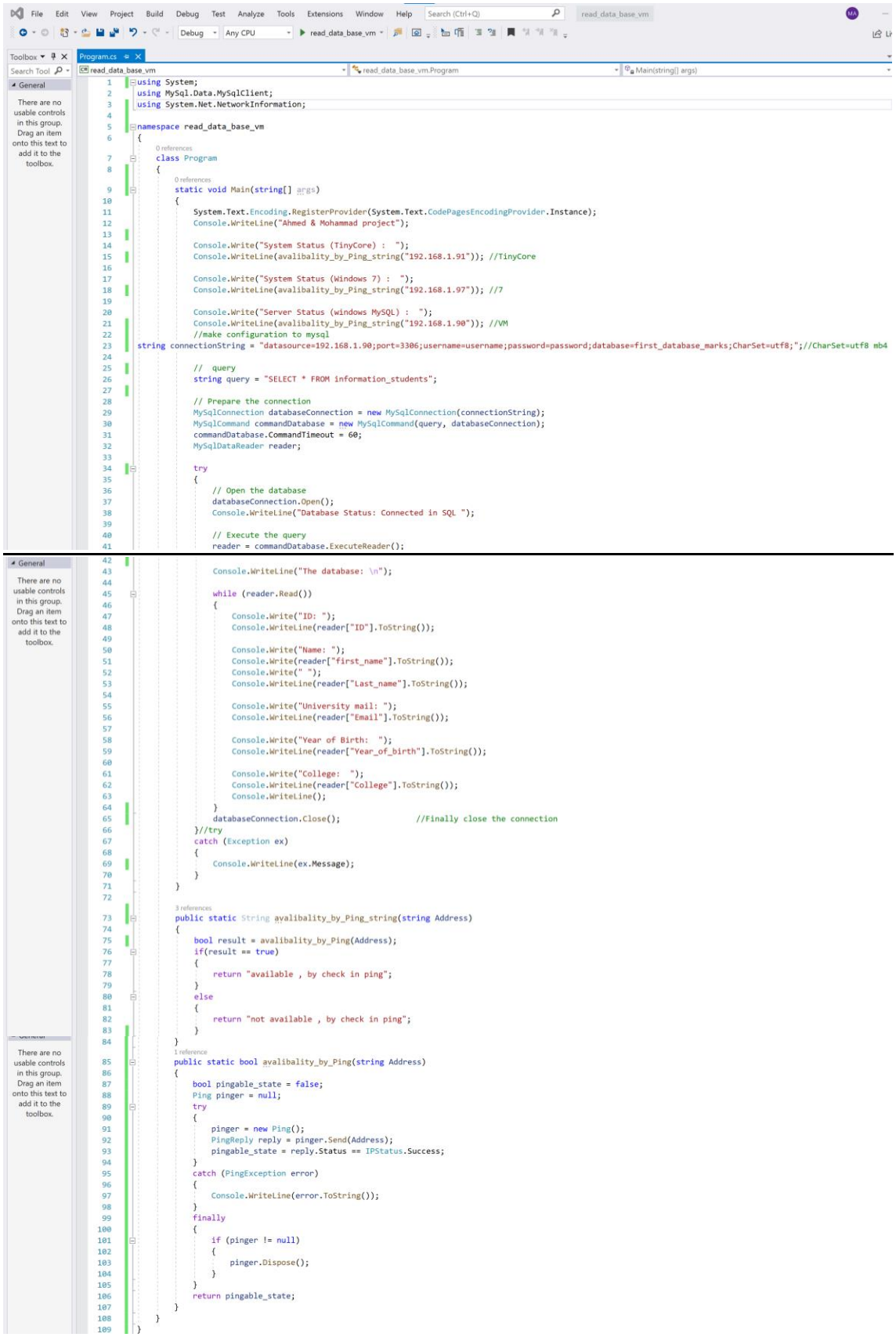
Appendix (A)

Availability Check Program by Ping:




```
using System;
using System.Net.NetworkInformation;
// C# program to check avalibality by Ping
namespace Network_avalibality_GP1 {
    class Program {
        static void Main(string[] args) {

            Console.WriteLine("Ahmed & Mohammad project");

            Console.WriteLine("Ahmed");
            Console.WriteLine(avalibality_by_Ping("192.168.1.102")); //Ahmed

            Console.WriteLine("Ali");
            Console.WriteLine(avalibality_by_Ping("192.168.1.101")); //Ali
    }
/*A function that checks the avalibality by sending a ping command
and comparing with replay status. */
public static bool avalibality_by_Ping(string Address)
{
    bool pingable_state = false;
    Ping pinger = null;
    try
    {
        pinger = new Ping();
        PingReply reply = pinger.Send(Address);
        pingable_state = reply.Status == IPStatus.Success;
    }
    catch (PingException error)
    {
        Console.WriteLine(error.ToString());
    }
    finally
    {
        if (pinger != null)
        {
            pinger.Dispose();
        }
    }
    return pingable_state;
}
    }
}
```

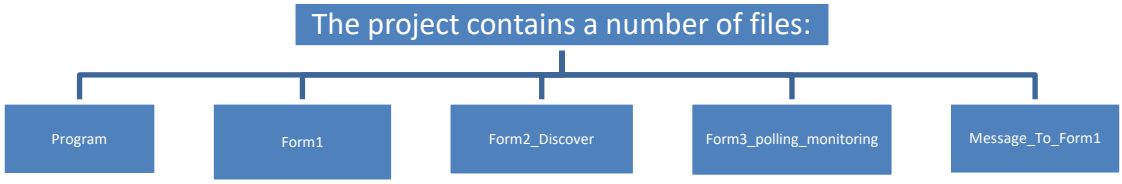
Database service checking program:



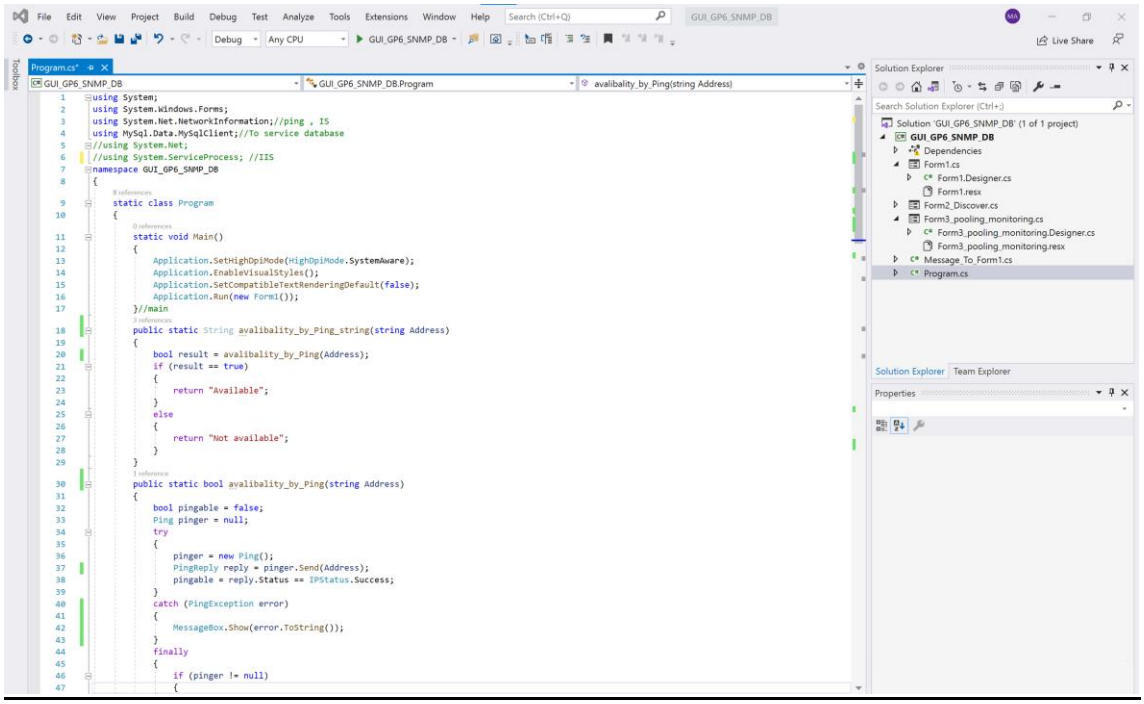
```
using System;
using MySql.Data.MySqlClient;
using System.Net.NetworkInformation;

namespace read_data_base_vm
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);
            Console.WriteLine("Ahmed & Mohammad project");

            Console.Write("System Status (TinyCore) :  ");
            Console.WriteLine(avalibality_by_Ping_string("192.168.1.91")); //TinyCore

            Console.Write("System Status (Windows 7) :  ");
            Console.WriteLine(avalibality_by_Ping_string("192.168.1.97")); //7

            Console.Write("Server Status (windows MySQL) :  ");
            Console.WriteLine(avalibality_by_Ping_string("192.168.1.90")); //VM
            //make configuration to mysql
string connectionString = "datasource=192.168.1.90;port=3306;username=username;password=password;database=first_database_marks;CharSet=utf8;";//CharSet=utf8 mb4

            //  query
            string query = "SELECT * FROM information_students";

            // Prepare the connection
            MySqlConnection databaseConnection = new MySqlConnection(connectionString);
            MySqlCommand commandDatabase = new MySqlCommand(query, databaseConnection);
            commandDatabase.CommandTimeout = 60;
            MySqlDataReader reader;

            try
            {
                // Open the database
                databaseConnection.Open();
                Console.WriteLine("Database Status: Connected in SQL ");

                // Execute the query
                reader = commandDatabase.ExecuteReader();

                Console.WriteLine("The database: \n");

                while (reader.Read())
                {
                    Console.Write("ID: ");
                    Console.WriteLine(reader["ID"].ToString());

                    Console.Write("Name: ");
                    Console.Write(reader["first_name"].ToString());
                    Console.Write(" ");
                    Console.WriteLine(reader["Last_name"].ToString());

                    Console.Write("University mail: ");
                    Console.WriteLine(reader["Email"].ToString());

                    Console.Write("Year of Birth:  ");
                    Console.WriteLine(reader["Year_of_birth"].ToString());

                    Console.Write("College:  ");
                    Console.WriteLine(reader["College"].ToString());
                    Console.WriteLine();
                }
                databaseConnection.Close();               //Finally close the connection
            }//try
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        public static String avalibality_by_Ping_string(string Address)
        {
            bool result = avalibality_by_Ping(Address);
            if(result == true)
            {
                return "available , by check in ping";
            }
            else
            {
                return "not available , by check in ping";
            }
        }
        public static bool avalibality_by_Ping(string Address)
        {
            bool pingable_state = false;
            Ping pinger = null;
            try
            {
                pinger = new Ping();
                PingReply reply = pinger.Send(Address);
                pingable_state = reply.Status == IPStatus.Success;
            }
            catch (PingException error)
            {
                Console.WriteLine(error.ToString());
            }
            finally
            {
                if (pinger != null)
                {
                    pinger.Dispose();
                }
            }
            return pingable_state;
        }
    }
}
```

Appendix (B)

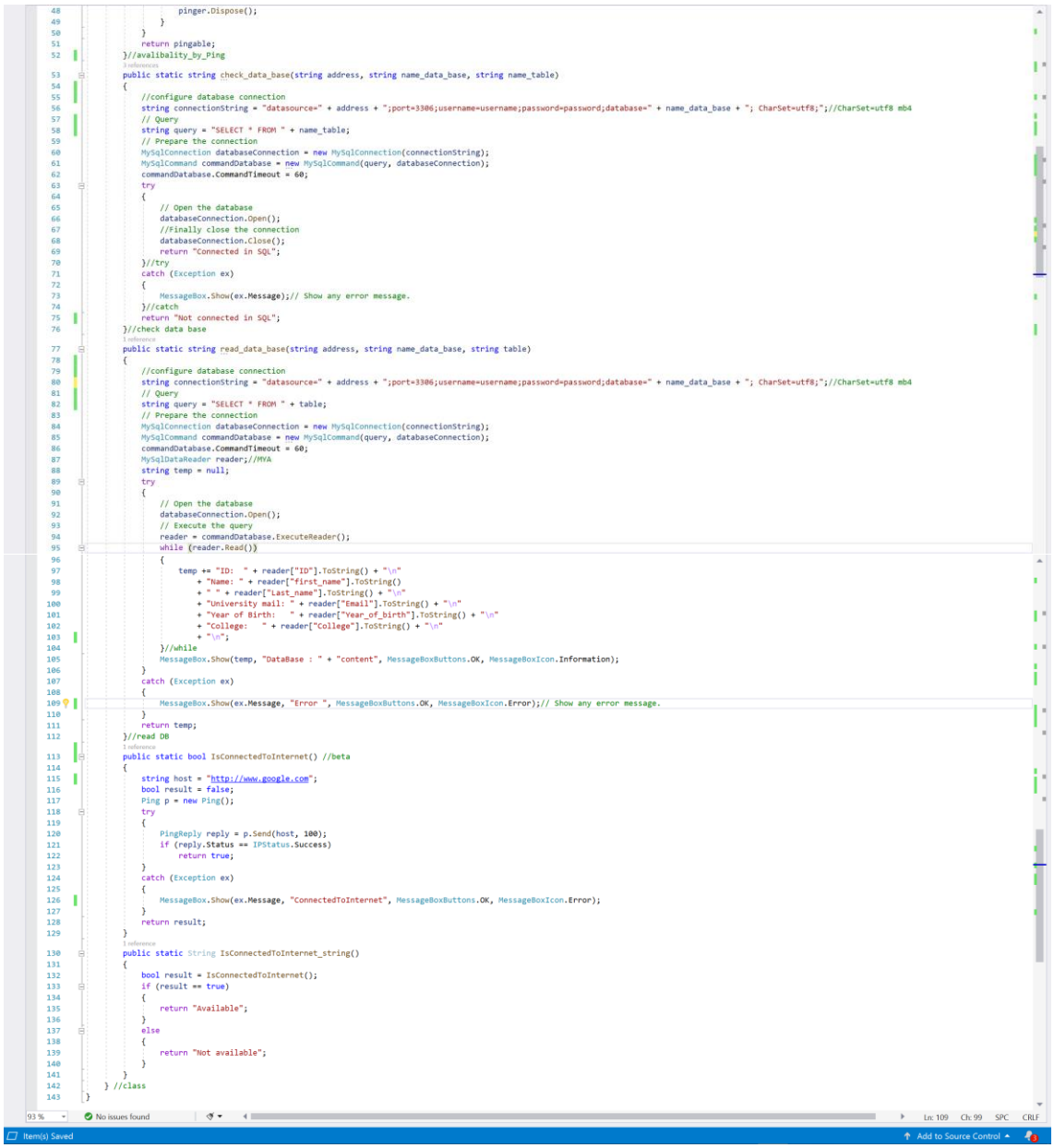
Programming with GUI:



Program:

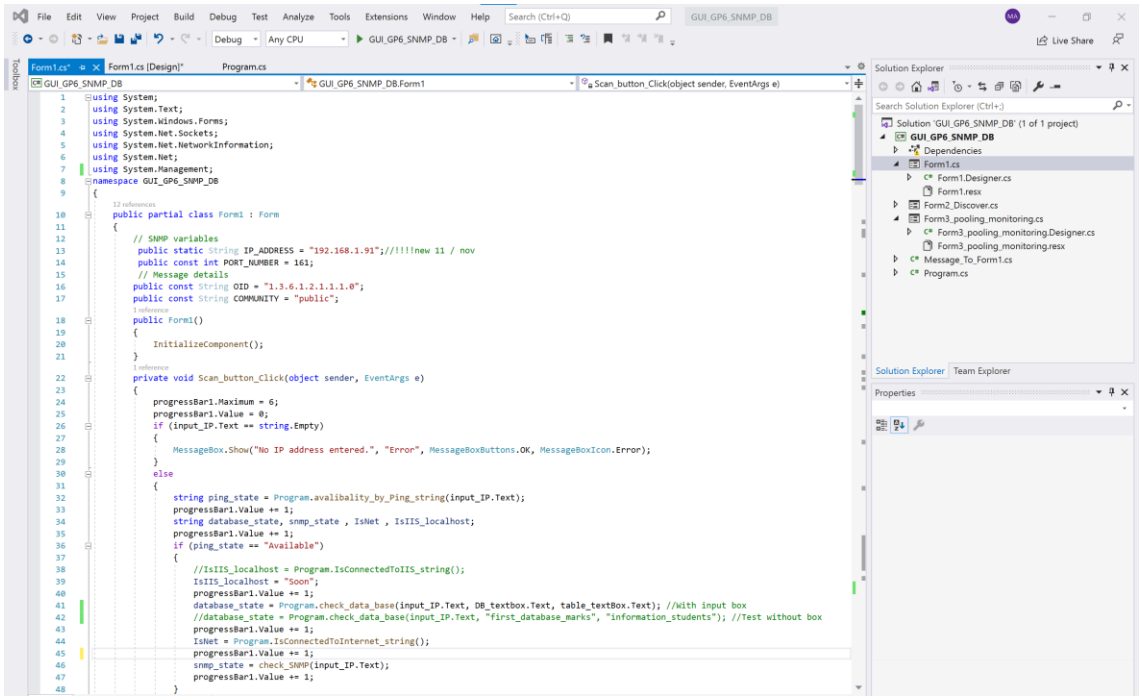


```
using System;
using System.Windows.Forms;
using System.Net.NetworkInformation;//ping , IS
using MySql.Data.MySqlClient;//To service database
//using System.Net;
//using System.ServiceProcess; //IIS
namespace GUI_GP6_SNMP_DB
{
    static class Program
    {
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }//main
        public static String avalibality_by_Ping_string(string Address)
        {
            bool result = avalibality_by_Ping(Address);
            if (result == true)
            {
                return "Available";
            }
            else
            {
                return "Not available";
            }
        }
        public static bool avalibality_by_Ping(string Address)
        {
            bool pingable = false;
            Ping pinger = null;
            try
            {
                pinger = new Ping();
                PingReply reply = pinger.Send(Address);
                pingable = reply.Status == IPStatus.Success;
            }
            catch (PingException error)
            {
                MessageBox.Show(error.ToString());
            }
            finally
            {
                if (pinger != null)
                {
```

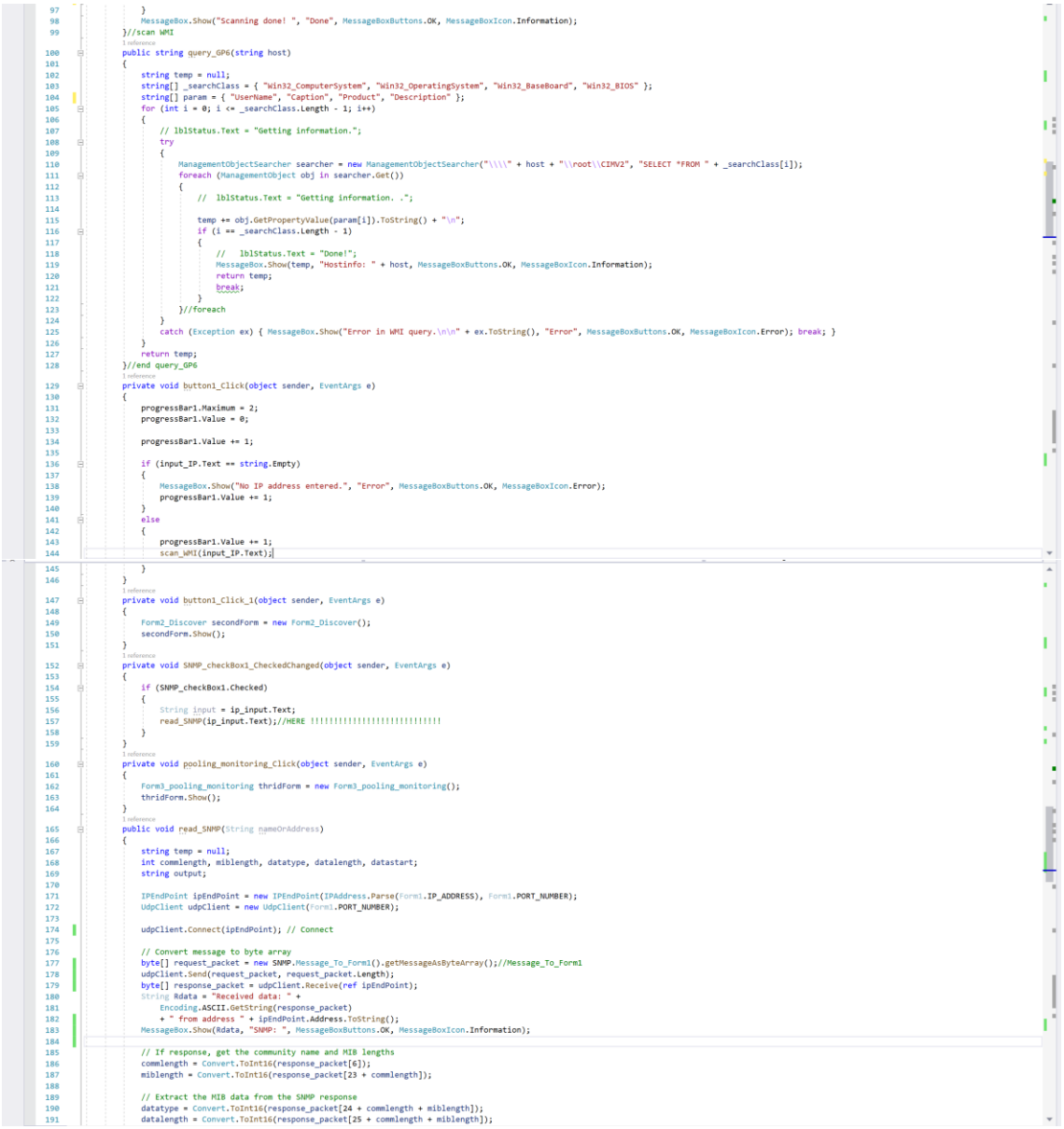


```
                pinger.Dispose();
            }
        }
        return pingable;
    }//avalibality_by_Ping
    public static string check_data_base(string address, string name_data_base, string name_table)
    {
        //configure database connection
        string connectionString = "datasource=" + address + ";port=3306;username=username;password=password;database=" + name_data_base + "; CharSet=utf8;";//CharSet=utf8 mb4
        // Query
        string query = "SELECT * FROM " + name_table;
        // Prepare the connection
        MySqlConnection databaseConnection = new MySqlConnection(connectionString);
        MySqlCommand commandDatabase = new MySqlCommand(query, databaseConnection);
        commandDatabase.CommandTimeout = 60;
        try
        {
            // Open the database
            databaseConnection.Open();
            //finally close the connection
            databaseConnection.Close();
            return "Connected in SQL";
        }//try
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);// Show any error message.
        }//catch
        return "Not connected in SQL";
    }//check data base
    public static string read_data_base(string address, string name_data_base, string table)
    {
        //configure database connection
        string connectionString = "datasource=" + address + ";port=3306;username=username;password=password;database=" + name_data_base + "; CharSet=utf8;";//CharSet=utf8 mb4
        // Query
        string query = "SELECT * FROM " + table;
        // Prepare the connection
        MySqlConnection databaseConnection = new MySqlConnection(connectionString);
        MySqlCommand commandDatabase = new MySqlCommand(query, databaseConnection);
        commandDatabase.CommandTimeout = 60;
        MySqlDataReader reader;//MYA
        string temp = null;
        try
        {
            // Open the database
            databaseConnection.Open();
            // Execute the query
            reader = commandDatabase.ExecuteReader();
            while (reader.Read())
            {
                temp += "ID:  " + reader["ID"].ToString() + "\n"
                    + "Name: " + reader["first_name"].ToString()
                    + " " + reader["last_name"].ToString() + "\n"
                    + "University mail: " + reader["Email"].ToString() + "\n"
                    + "Year of Birth:   " + reader["Year_of_birth"].ToString() + "\n"
                    + "College:   " + reader["College"].ToString() + "\n"
                    + "\n";
            }//while
            MessageBox.Show(temp, "DataBase : " + "content", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Error ", MessageBoxButtons.OK, MessageBoxIcon.Error);// Show any error message.
        }
        return temp;
    }//read DB
    public static bool IsConnectedToInternet() //beta
    {
        string host = "http://www.google.com";
        bool result = false;
        Ping p = new Ping();
        try
        {
            PingReply reply = p.Send(host, 100);
            if (reply.Status == IPStatus.Success)
                return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "ConnectedToInternet", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        return result;
    }
    public static String IsConnectedToInternet_string()
    {
        bool result = IsConnectedToInternet();
        if (result == true)
        {
            return "Available";
        }
        else
        {
            return "Not available";
        }
    }
} //class
}
```

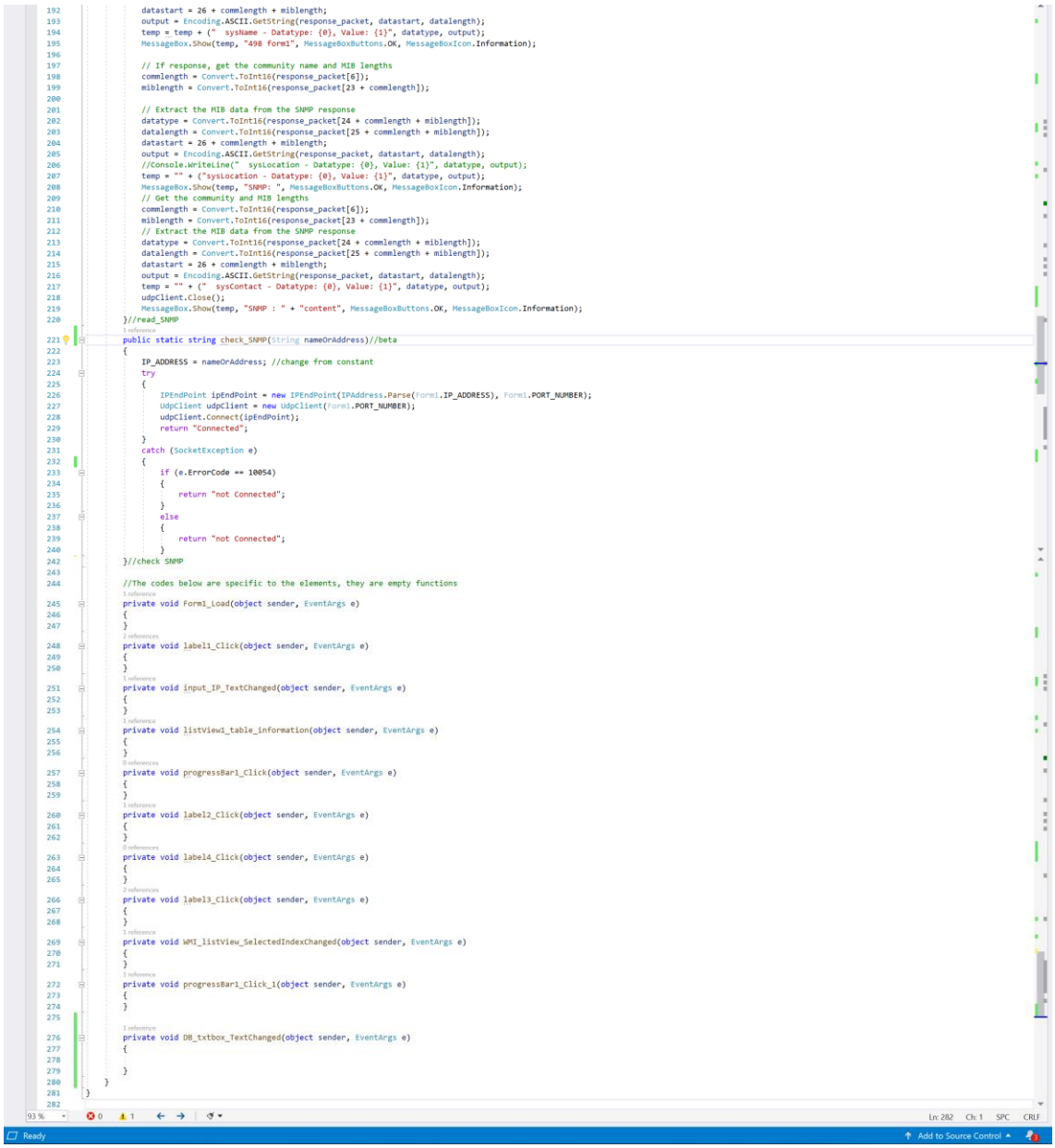
Form1:



```
using System;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net.NetworkInformation;
using System.Net;
using System.Management;
namespace GUI_GP6_SNMP_DB
{
    12 references
    public partial class Form1 : Form
    {
        // SNMP variables
         public static String IP_ADDRESS = "192.168.1.91";//!!!!new 11 / nov
         public const int PORT_NUMBER = 161;
         // Message details
        public const String OID = "1.3.6.1.2.1.1.1.0";
        public const String COMMUNITY = "public";
        1 reference
        public Form1()
        {
            InitializeComponent();
        }
        1 reference
        private void Scan_button_Click(object sender, EventArgs e)
        {
            progressBar1.Maximum = 6;
            progressBar1.Value = 0;
            if (input_IP.Text == string.Empty)
            {
                MessageBox.Show("No IP address entered.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                string ping_state = Program.avalibality_by_Ping_string(input_IP.Text);
                progressBar1.Value += 1;
                string database_state, snmp_state , IsNet , IsIIS_localhost;
                progressBar1.Value += 1;
                if (ping_state == "Available")
                {
                    //IsIIS_localhost = Program.IsConnectedToIIS_string();
                    IsIIS_localhost = "Soon";
                    progressBar1.Value += 1;
                    database_state = Program.check_data_base(input_IP.Text, DB_textbox.Text, table_textBox.Text); //With input box
                    //database_state = Program.check_data_base(input_IP.Text, "first_database_marks", "information_students"); //Test without box
                    progressBar1.Value += 1;
                    IsNet = Program.IsConnectedToInternet_string();
                    progressBar1.Value += 1;
                    snmp_state = check_SNMP(input_IP.Text);
                    progressBar1.Value += 1;
                }
                else
                {
                    progressBar1.Value += 1;
                    IsIIS_localhost = "Not connected in localhost";
                    progressBar1.Value += 1;
                    database_state = "Not connected in SQL";
                    snmp_state = "Not Connected";
                    progressBar1.Value += 1;
                    IsNet = "Not Connected";
                    progressBar1.Value += 1;
                }
                listVAddr.Items.Add(new ListViewItem(new String[] { input_IP.Text , ping_state , IsNet, IsIIS_localhost, snmp_state, database_state}));
                MessageBox.Show("Scanning done! ", "Done", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
        1 reference
        private void Stop_button_Click(object sender, EventArgs e)
        {
            Stop_button.Enabled = true;
            Stop_button.Enabled = false;
            input_IP.Enabled = true;
        }
        1 reference
        private void read_DB_checkBox_Changed(object sender, EventArgs e)
        {
            if (read_DB_checkBox.Checked)
            {
                string database_state = Program.check_data_base(input_IP.Text, DB_textbox.Text, table_textBox.Text); //With input box
                //string database_state = Program.check_data_base(input_IP.Text, "first_database_marks", "information_students"); //Test without box
            }
        }
        1 reference
        private void scan_WMI(string ip_address)
        {
            Ping myPing;
            PingReply reply;
            IPAddress addr;
            IPHostEntry host;
            myPing = new Ping();
            reply = myPing.Send(ip_address);

            if (reply.Status == IPStatus.Success)
                {
                    try
                    {
                        addr = IPAddress.Parse(ip_address);
                        host = Dns.GetHostEntry(addr);
                        string info_query_GP6 = query_GP6(ip_address);
                        WMIlistView.Items.Add(new ListViewItem(new String[] { ip_address, host.HostName, "Up" , info_query_GP6 , "soon"}));
                    }
                    catch { MessageBox.Show("Couldnt retrieve hostname for " + ip_address, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }
```

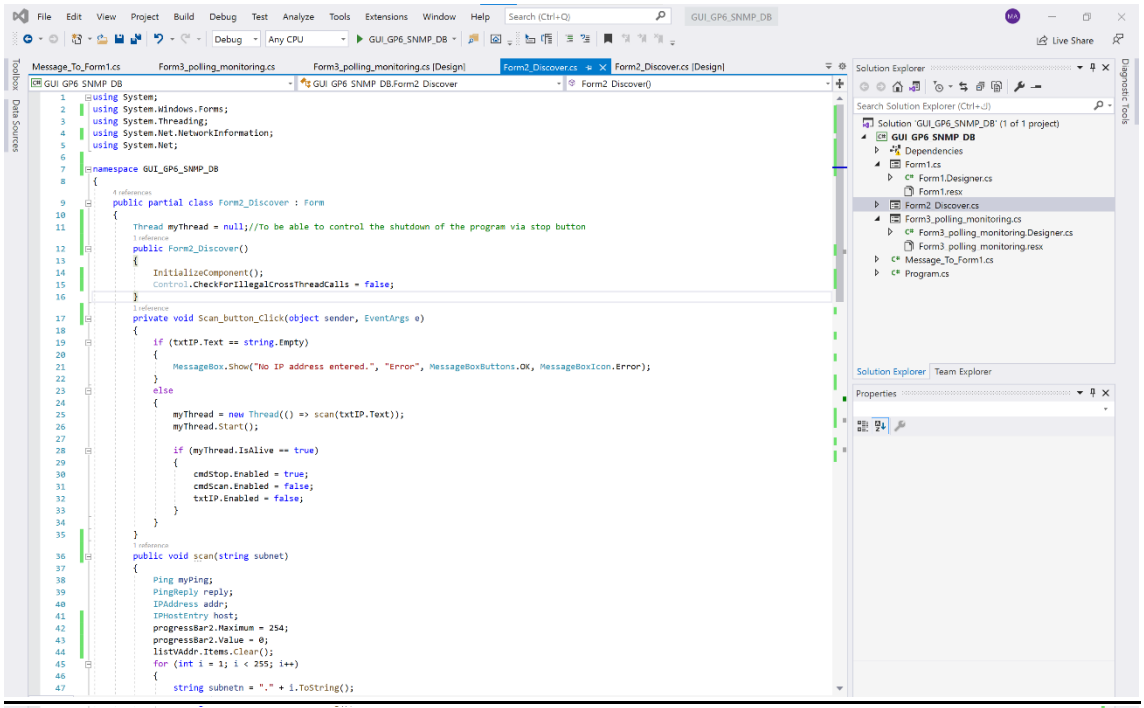


```
            }
            MessageBox.Show("Scanning done! ", "Done", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }//scan WMI
        1 reference
        public string query_GP6(string host)
        {
            string temp = null;
            string[] _searchClass = { "Win32_ComputerSystem", "Win32_OperatingSystem", "Win32_BaseBoard", "Win32_BIOS" };
            string[] param = { "UserName", "Caption", "Product", "Description" };
            for (int i = 0; i <= _searchClass.Length - 1; i++)
            {
                // lblStatus.Text = "Getting information.";
                try
                {
                    ManagementObjectSearcher searcher = new ManagementObjectSearcher("\\\\" + host + "\\root\\CIMV2", "SELECT *FROM " + _searchClass[i]);
                    foreach (ManagementObject obj in searcher.Get())
                    {
                        //  lblStatus.Text = "Getting information. .";

                        temp += obj.GetPropertyValue(param[i]).ToString() + "\n";
                        if (i == _searchClass.Length - 1)
                        {
                            //   lblStatus.Text = "Done!";
                            MessageBox.Show(temp, "Hostinfo: " + host, MessageBoxButtons.OK, MessageBoxIcon.Information);
                            return temp;
                            break;
                        }
                    }//foreach
                }
                catch (Exception ex) { MessageBox.Show("Error in WMI query.\n\n" + ex.ToString(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); break; }
            }
            return temp;
        }//end query_GP6
        1 reference
        private void button1_Click(object sender, EventArgs e)
        {
            progressBar1.Maximum = 2;
            progressBar1.Value = 0;

            progressBar1.Value += 1;

            if (input_IP.Text == string.Empty)
            {
                MessageBox.Show("No IP address entered.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                progressBar1.Value += 1;
            }
            else
            {
                progressBar1.Value += 1;
                scan_WMI(input_IP.Text);
            }
        }
        1 reference
        private void button1_Click_1(object sender, EventArgs e)
        {
            Form2_Discover secondForm = new Form2_Discover();
            secondForm.Show();
        }
        1 reference
        private void SNMP_checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (SNMP_checkBox1.Checked)
            {
                String input = ip_input.Text;
                read_SNMP(ip_input.Text);//HERE !!!!!!!!!!!!!!!!!!!!!!!!!!!!
            }
        }
        1 reference
        private void pooling_monitoring_Click(object sender, EventArgs e)
        {
            Form3_pooling_monitoring thridForm = new Form3_pooling_monitoring();
            thridForm.Show();
        }
        1 reference
        public void read_SNMP(String nameOrAddress)
        {
            string temp = null;
            int commlength, miblength, datatype, datalength, datastart;
            string output;

            IPEndPoint ipEndPoint = new IPEndPoint(IPAddress.Parse(Form1.IP_ADDRESS), Form1.PORT_NUMBER);
            UdpClient udpClient = new UdpClient(Form1.PORT_NUMBER);

            udpClient.Connect(ipEndPoint); // Connect

            // Convert message to byte array
            byte[] request_packet = new SNMP.Message_To_Form1().getMessageAsByteArray();//Message_To_Form1
            udpClient.Send(request_packet, request_packet.Length);
            byte[] response_packet = udpClient.Receive(ref ipEndPoint);
            String Rdata = "Received data: " +
                Encoding.ASCII.GetString(response_packet)
                + " from address " + ipEndPoint.Address.ToString();
            MessageBox.Show(Rdata, "SNMP: ", MessageBoxButtons.OK, MessageBoxIcon.Information);

            // If response, get the community name and MIB lengths
            commlength = Convert.ToInt16(response_packet[6]);
            miblength = Convert.ToInt16(response_packet[23 + commlength]);

            // Extract the MIB data from the SNMP response
            datatype = Convert.ToInt16(response_packet[24 + commlength + miblength]);
            datalength = Convert.ToInt16(response_packet[25 + commlength + miblength]);
```



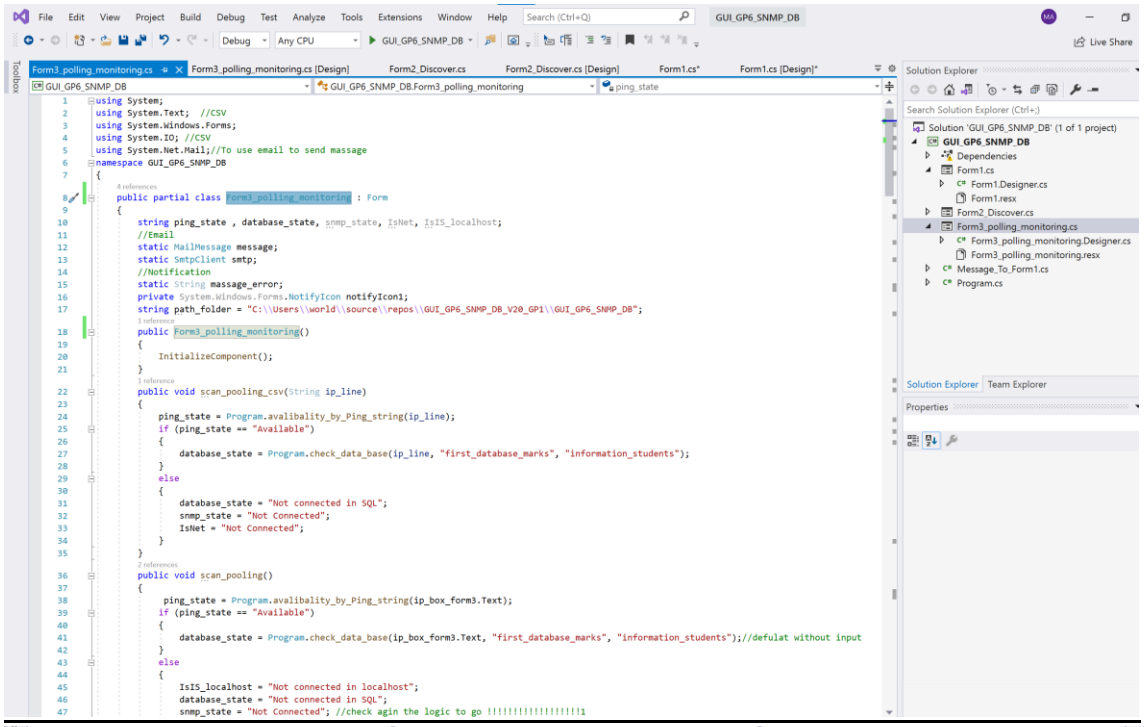
```
            datastart = 26 + commlength + miblength;
            output = Encoding.ASCII.GetString(response_packet, datastart, datalength);
            temp = temp + ("  sysName - Datatype: {0}, Value: {1}", datatype, output);
            MessageBox.Show(temp, "498 form1", MessageBoxButtons.OK, MessageBoxIcon.Information);

            // If response, get the community name and MIB lengths
            commlength = Convert.ToInt16(response_packet[6]);
            miblength = Convert.ToInt16(response_packet[23 + commlength]);

            // Extract the MIB data from the SNMP response
            datatype = Convert.ToInt16(response_packet[24 + commlength + miblength]);
            datalength = Convert.ToInt16(response_packet[25 + commlength + miblength]);
            datastart = 26 + commlength + miblength;
            output = Encoding.ASCII.GetString(response_packet, datastart, datalength);
            //Console.WriteLine("  sysLocation - Datatype: {0}, Value: {1}", datatype, output);
            temp = "" + ("sysLocation - Datatype: {0}, Value: {1}", datatype, output);
            MessageBox.Show(temp, "SNMP: ", MessageBoxButtons.OK, MessageBoxIcon.Information);
            // Get the community and MIB lengths
            commlength = Convert.ToInt16(response_packet[6]);
            miblength = Convert.ToInt16(response_packet[23 + commlength]);
            // Extract the MIB data from the SNMP response
            datatype = Convert.ToInt16(response_packet[24 + commlength + miblength]);
            datalength = Convert.ToInt16(response_packet[25 + commlength + miblength]);
            datastart = 26 + commlength + miblength;
            output = Encoding.ASCII.GetString(response_packet, datastart, datalength);
            temp = "" + ("  sysContact - Datatype: {0}, Value: {1}", datatype, output);
            udpClient.Close();
            MessageBox.Show(temp, "SNMP : " + "content", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }//read_SNMP
        1 reference
        public static string check_SNMP(String nameOrAddress)//beta
        {
            IP_ADDRESS = nameOrAddress; //change from constant
            try
            {
                IPEndPoint ipEndPoint = new IPEndPoint(IPAddress.Parse(Form1.IP_ADDRESS), Form1.PORT_NUMBER);
                UdpClient udpClient = new UdpClient(Form1.PORT_NUMBER);
                udpClient.Connect(ipEndPoint);
                return "Connected";
            }
            catch (SocketException e)
            {
                if (e.ErrorCode == 10054)
                {
                    return "not Connected";
                }
                else
                {
                    return "not Connected";
                }
        }//check SNMP

        //The codes below are specific to the elements, they are empty functions
        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {
        }
        2 references
        private void label1_Click(object sender, EventArgs e)
        {
        }
        1 reference
        private void input_IP_TextChanged(object sender, EventArgs e)
        {
        }
        1 reference
        private void listView1_table_information(object sender, EventArgs e)
        {
        }
        0 references
        private void progressBar1_Click(object sender, EventArgs e)
        {
        }
        1 reference
        private void label2_Click(object sender, EventArgs e)
        {
        }
        0 references
        private void label4_Click(object sender, EventArgs e)
        {
        }
        2 references
        private void label3_Click(object sender, EventArgs e)
        {
        }
        1 reference
        private void WMI_listView_SelectedIndexChanged(object sender, EventArgs e)
        {
        }
        1 reference
        private void progressBar1_Click_1(object sender, EventArgs e)
        {
        }

        1 reference
        private void DB_txtbox_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

Form2_Discover:



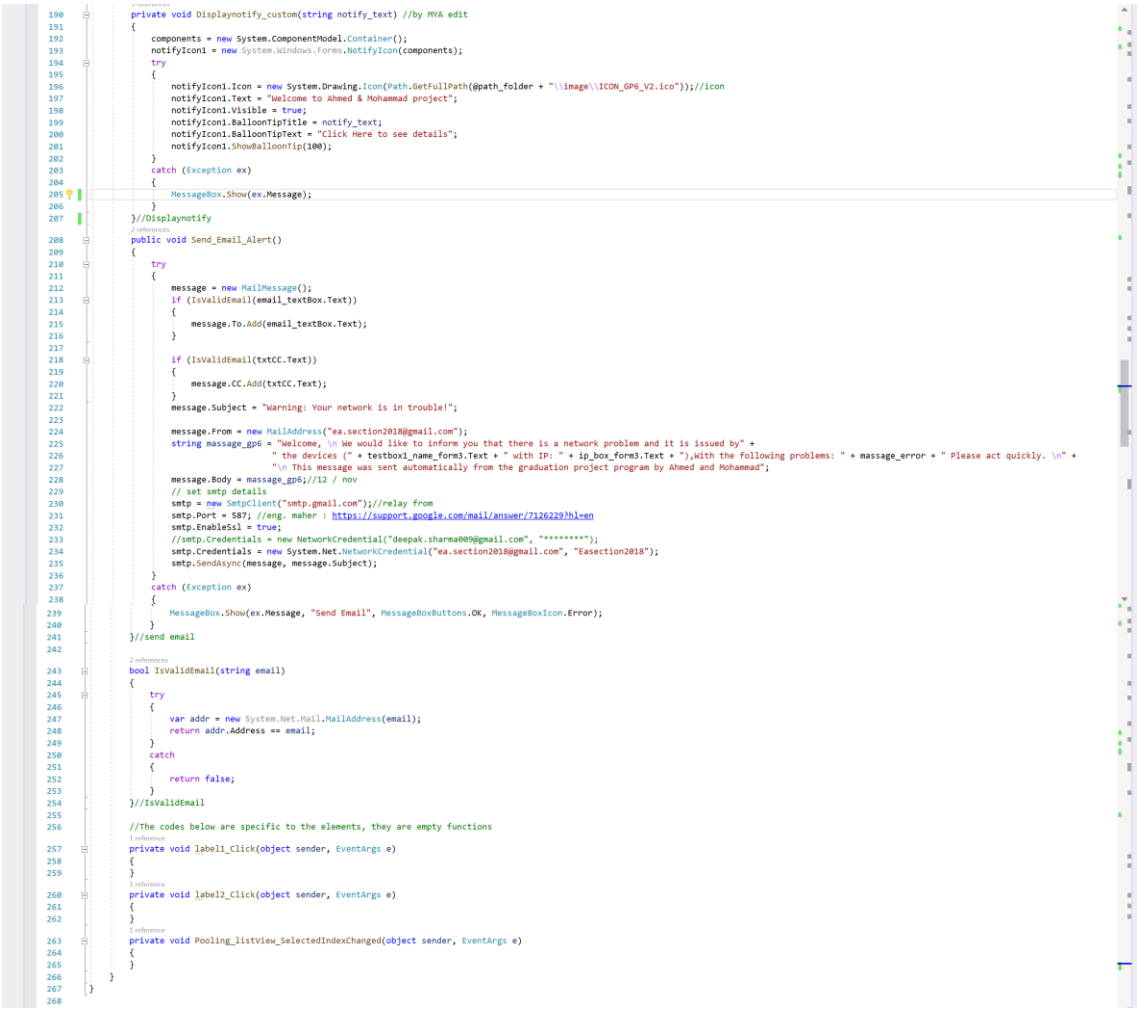
```
using System;
using System.Windows.Forms;
using System.Threading;
using System.Net.NetworkInformation;
using System.Net;

namespace GUI_GP6_SNMP_DB
{
    public partial class Form2_Discover : Form
    {
        Thread myThread = null;//To be able to control the shutdown of the program via stop button
        public Form2_Discover()
        {
            InitializeComponent();
            Control.CheckForIllegalCrossThreadCalls = false;
        }
        private void Scan_button_Click(object sender, EventArgs e)
        {
            if (txtIP.Text == string.Empty)
            {
                MessageBox.Show("No IP address entered.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                myThread = new Thread(() => scan(txtIP.Text));
                myThread.Start();

                if (myThread.IsAlive == true)
                {
                    cmdStop.Enabled = true;
                    cmdScan.Enabled = false;
                    txtIP.Enabled = false;
                }
            }
        }
        public void scan(string subnet)
        {
            Ping myPing;
            PingReply reply;
            IPAddress addr;
            IPHostEntry host;
            progressBar2.Maximum = 254;
            progressBar2.Value = 0;
            listVAddr.Items.Clear();
            for (int i = 1; i < 255; i++)
            {
                string subnetn = "." + i.ToString();
                myPing = new Ping();
                reply = myPing.Send(subnet + subnetn, 900);

                if (reply.Status == IPStatus.Success)
                {
                    try
                    {
                        addr = IPAddress.Parse(subnet + subnetn);
                        host = Dns.GetHostEntry(addr);
                        listVAddr.Items.Add(new ListViewItem(new String[] { subnet + subnetn, host.HostName, "Up" }));
                    }
                    catch { MessageBox.Show("Couldnt retrieve hostname for " + subnet + subnetn, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error); }
                }
                progressBar2.Value += 1;
            }
            cmdScan.Enabled = true;
            cmdStop.Enabled = false;
            txtIP.Enabled = true;
            int count = listVAddr.Items.Count;
            MessageBox.Show("Scanning done!\nFound " + count.ToString() + " hosts.", "Done", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        private void Stop_button_Click(object sender, EventArgs e)
        {
            myThread.Suspend();
            cmdScan.Enabled = true;
            cmdStop.Enabled = false;
            txtIP.Enabled = true;
        }
        private void listView1_SelectedIndexChanged(object sender, EventArgs e)
        {
        }

        private void Form2_Discover_Load(object sender, EventArgs e)
        {
        }
        private void input_IP_TextChanged(object sender, EventArgs e)
        {
        }
    }
}
```

Form3_polling_monitoring:



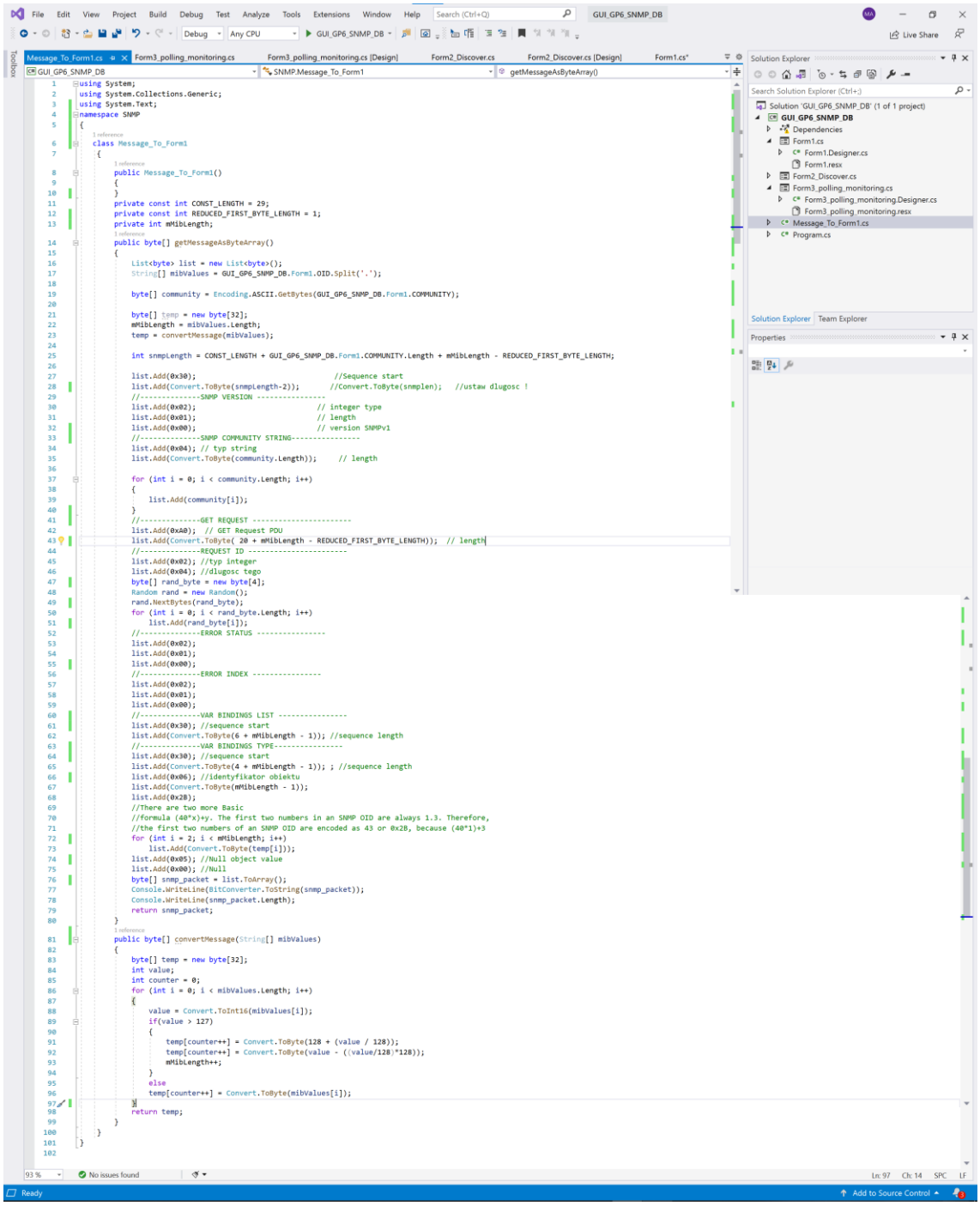
```
using System;
using System.Text;  //CSV
using System.Windows.Forms;
using System.IO; //CSV
using System.Net.Mail;//To use email to send massage
namespace GUI_GP6_SNMP_DB
{
    4 references
    public partial class Form3_polling_monitoring : Form
    {
        string ping_state , database_state, snmp_state, IsNet, IsIS_localhost;
        //Email
        static MailMessage message;
        static SmtpClient smtp;
        //Notification
        static String massage_error;
        private System.Windows.Forms.NotifyIcon notifyIcon1;
        string path_folder = "C:\\Users\\world\\source\\repos\\GUI_GP6_SNMP_DB_V20_GP1\\GUI_GP6_SNMP_DB";
        1 reference
        public Form3_polling_monitoring()
        {
            InitializeComponent();
        }
        1 reference
        public void scan_pooling_csv(String ip_line)
        {
            ping_state = Program.avalibality_by_Ping_string(ip_line);
            if (ping_state == "Available")
            {
                database_state = Program.check_data_base(ip_line, "first_database_marks", "information_students");
            }
            else
            {
                database_state = "Not connected in SQL";
                snmp_state = "Not Connected";
                IsNet = "Not Connected";
            }
        }
        2 references
        public void scan_pooling()
        {
             ping_state = Program.avalibality_by_Ping_string(ip_box_form3.Text);
            if (ping_state == "Available")
            {
                database_state = Program.check_data_base(ip_box_form3.Text, "first_database_marks", "information_students");//defulat without input
            }
            else
            {
                IsIS_localhost = "Not connected in localhost";
                database_state = "Not connected in SQL";
                snmp_state = "Not Connected"; //check agin the logic to go !!!!!!!!!!!!!!!!!1
                IsNet = "Not Connected";
            }
        }
        1 reference
        private void save_Click(object sender, EventArgs e)
        {
            Displaynotify(); //test
            write_csv(testbox1_name_form3.Text, ip_box_form3.Text);
            scan_pooling();
            Pooling_listView.Items.Add(new ListViewItem(new String[] { testbox1_name_form3.Text, ip_box_form3.Text, ping_state, "soon" , database_state }));
        }
        //CSV
        1 reference
        private void write_csv(string name, string ip)
        {
            string filePath = @path_folder + "\\IP_database.csv";
            string delimiter = ",";
            string[][] output = new string[][]{
                new string[]{ name , ip }
            };
            int length = output.GetLength(0);
            StringBuilder sb = new StringBuilder();
            for (int index = 0; index < length; index++)
                sb.AppendLine(string.Join(delimiter, output[index]));
            File.AppendAllText(filePath, sb.ToString());
        }
        //read_csv3_automatic
        1 reference
        private void read_csv_automatic()//edit with group to auto :) 15/nov/2020
        {
            int count = 0;
            try
            {
                string path =   @path_folder + "\\IP_database.csv";//Name of file
                var lines = File.ReadAllLines(path);
                foreach (string line in lines)
                {
                    count = count + 1;
                    var parts = line.Split(',');
                    ListViewItem lvi = new ListViewItem(parts[0]);//parts[0] Name
                    scan_pooling_csv(parts[1]);
                    Pooling_listView.Items.Add(new ListViewItem(new String[] { parts[0] , parts[1],  ping_state, "soon", database_state }));
                }
                MessageBox.Show("From csv file: " + (count + " rows imported.")); // update count

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

```
1 reference
private void read_csv_manual()
{
    OpenFileDialog ofDialog = new OpenFileDialog();
    ofDialog.Filter = @"CSV files|*.csv";
    ofDialog.Title = @"Select your backlink file...";
    ofDialog.FileName = "backlinks.csv";
    int count = 0;
    if (ofDialog.ShowDialog() == DialogResult.Cancel)           // is cancel pressed?
        return;
    try
    {
        string filename = ofDialog.FileName;
        var lines = File.ReadAllLines(filename);
        foreach (string line in lines)
        {
            count = count + 1;
            var parts = line.Split(',');
            ListViewItem lvi = new ListViewItem(parts[0]);
            lvi.SubItems.Add(parts[1]);
            Pooling_listView.Items.Add(lvi);
        }
        MessageBox.Show("From csv file: " + (count  + " rows imported.");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
1 reference
private void Form3_pooling_monitoring_Load(object sender, EventArgs e)
{
    read_csv_automatic();
}
2 references
private void Displaynotify()
{
    components = new System.ComponentModel.Container();
    notifyIcon1 = new System.Windows.Forms.NotifyIcon(components);
    try
    {
        notifyIcon1.Icon = new System.Drawing.Icon(Path.GetFullPath(@path_folder + "\\image\\ICON_GP6_V2.ico"));//icon
        notifyIcon1.Text = "Welcome to Ahmed & Mohammad project";
        notifyIcon1.Visible = true;
        notifyIcon1.BalloonTipTitle = "Welcome";
        notifyIcon1.BalloonTipText = "Click Here to see details";
        notifyIcon1.ShowBalloonTip(100);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}//Displaynotify
1 reference
private void load_Click(object sender, EventArgs e)
{
    Displaynotify();
    read_csv_manual();
}
0 references
private void button1_Click(object sender, EventArgs e)
{
    Send_Email_Alert();
}
1 reference
private void Start_Pooling_Click(object sender, EventArgs e)
{
    int counter_to_email = 0;
    bool oneTimes = true; //mya
    while (true)
    {
        System.Threading.Thread.Sleep(5000); //5s
        scan_pooling();
        if(ping_state == "Not available" && database_state == "Not connected in SQL")//AND && from Ahmed idea 16/nov/2020
        {
            massage_error = "We lost the connection to the device (unavailable) and database";
            Displaynotify_custom(massage_error);
            counter_to_email = counter_to_email + 1;
        }
        else if (ping_state == "Not available")
        {
            massage_error = "We lost the connection to the device (unavailable)";
            Displaynotify_custom(massage_error);
            counter_to_email = counter_to_email + 1;
        }
        else if (database_state == "Not connected in SQL")
            {
            massage_error = "We lost the connection in the database";
            Displaynotify_custom(massage_error);
            counter_to_email = counter_to_email + 1;
        }
        //email
        if(counter_to_email > 5 && oneTimes == true)
        {
            Send_Email_Alert();
            MessageBox.Show("send email");
            oneTimes = false;
        }
    }
}//start pooling
```



```
        private void Displaynotify_custom(string notify_text) //by MYA edit
        {
            components = new System.ComponentModel.Container();
            notifyIcon1 = new System.Windows.Forms.NotifyIcon(components);
            try
            {
                notifyIcon1.Icon = new System.Drawing.Icon(Path.GetFullPath(@path_folder + "\\image\\ICON_GP6_V2.ico"));//icon
                notifyIcon1.Text = "Welcome to Ahmed & Mohammad project";
                notifyIcon1.Visible = true;
                notifyIcon1.BalloonTipTitle = notify_text;
                notifyIcon1.BalloonTipText = "Click Here to see details";
                notifyIcon1.ShowBalloonTip(100);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }//Displaynotify
        2 references
        public void Send_Email_Alert()
        {
            try
            {
                message = new MailMessage();
                if (IsValidEmail(email_textBox.Text))
                {
                    message.To.Add(email_textBox.Text);
                }

                if (IsValidEmail(txtCC.Text))
                {
                    message.CC.Add(txtCC.Text);
                }
                message.Subject = "Warning: Your network is in trouble!";

                message.From = new MailAddress("ea.section2018@gmail.com");
                string massage_gp6 = "Welcome, \n We would like to inform you that there is a network problem and it is issued by" +
                                     " the devices (" + testbox1_name_form3.Text + " with IP: " + ip_box_form3.Text + "),With the following problems: " + massage_error + " Please act quickly. \n" +
                                     "\n This message was sent automatically from the graduation project program by Ahmed and Mohammad";
                message.Body = massage_gp6;//12 / nov
                // set smtp details
                smtp = new SmtpClient("smtp.gmail.com");//relay from
                smtp.Port = 587; //eng. maher : https://support.google.com/mail/answer/7126229?hl=en
                smtp.EnableSsl = true;
                //smtp.Credentials = new NetworkCredential("deepak.sharma009@gmail.com", "*********");
                smtp.Credentials = new System.Net.NetworkCredential("ea.section2018@gmail.com", "Easection2018");
                smtp.SendAsync(message, message.Subject);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Send Email", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }//send email

        2 references
        bool IsValidEmail(string email)
        {
            try
            {
                var addr = new System.Net.Mail.MailAddress(email);
                return addr.Address == email;
            }
            catch
            {
                return false;
            }
        }//IsValidEmail

        //The codes below are specific to the elements, they are empty functions
        1 reference
        private void label1_Click(object sender, EventArgs e)
        {
        }
        1 reference
        private void label2_Click(object sender, EventArgs e)
        {
        }
        1 reference
        private void Pooling_listView_SelectedIndexChanged(object sender, EventArgs e)
        {
        }
    }
}
```

Message_To_Form1:

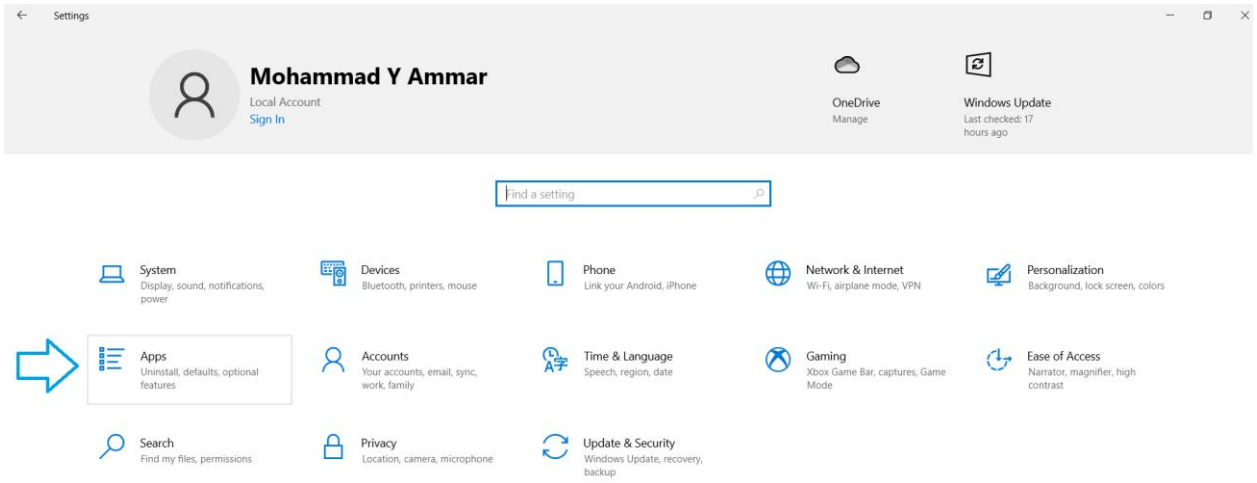


```
using System;
using System.Collections.Generic;
using System.Text;
namespace SNMP
{
    class Message_To_Form1
    {
        public Message_To_Form1()
        {
        }
        private const int CONST_LENGTH = 29;
        private const int REDUCED_FIRST_BYTE_LENGTH = 1;
        private int mMibLength;
        public byte[] getMessageAsByteArray()
        {
            List<byte> list = new List<byte>();
            String[] mibValues = GUI_GP6_SNMP_DB.Form1.OID.Split('.');

            byte[] community = Encoding.ASCII.GetBytes(GUI_GP6_SNMP_DB.Form1.COMMUNITY);

            byte[] temp = new byte[32];
            mMibLength = mibValues.Length;
            temp = convertMessage(mibValues);

            int snmpLength = CONST_LENGTH + GUI_GP6_SNMP_DB.Form1.COMMUNITY.Length + mMibLength - REDUCED_FIRST_BYTE_LENGTH;

            list.Add(0x30);                             //Sequence start
            list.Add(Convert.ToByte(snmpLength-2));     //Convert.ToByte(snmplen);   //ustaw dlugosc !
            //-------------SNMP VERSION ---------------
            list.Add(0x02);                             // integer type
            list.Add(0x01);                             // length
            list.Add(0x00);                             // version SNMPv1
            //-------------SNMP COMMUNITY STRING---------------
            list.Add(0x04); // typ string
            list.Add(Convert.ToByte(community.Length));     // length

            for (int i = 0; i < community.Length; i++)
            {
                list.Add(community[i]);
            }
            //-------------GET REQUEST ----------------------
            list.Add(0xA0);  // GET Request PDU
            list.Add(Convert.ToByte( 20 + mMibLength - REDUCED_FIRST_BYTE_LENGTH));  // length
            //-------------REQUEST ID ----------------------
            list.Add(0x02); //typ integer
            list.Add(0x04); //dlugosc tego
            byte[] rand_byte = new byte[4];
            Random rand = new Random();
            rand.NextBytes(rand_byte);
            for (int i = 0; i < rand_byte.Length; i++)
                list.Add(rand_byte[i]);
            //-------------ERROR STATUS ---------------
            list.Add(0x02);
            list.Add(0x01);
            list.Add(0x00);
            //-------------ERROR INDEX ---------------
            list.Add(0x02);
            list.Add(0x01);
            list.Add(0x00);
            //-------------VAR BINDINGS LIST ---------------
            list.Add(0x30); //sequence start
            list.Add(Convert.ToByte(6 + mMibLength - 1)); //sequence length
            //-------------VAR BINDINGS TYPE---------------
            list.Add(0x30); //sequence start
            list.Add(Convert.ToByte(4 + mMibLength - 1)); ; //sequence length
            list.Add(0x06); //identyfikator obiektu
            list.Add(Convert.ToByte(mMibLength - 1));
            list.Add(0x2B);
            //There are two more Basic
            //formula (40*x)+y. The first two numbers in an SNMP OID are always 1.3. Therefore,
            //the first two numbers of an SNMP OID are encoded as 43 or 0x2B, because (40*1)+3
            for (int i = 2; i < mMibLength; i++)
                list.Add(Convert.ToByte(temp[i]));
            list.Add(0x05); //Null object value
            list.Add(0x00); //Null
            byte[] snmp_packet = list.ToArray();
            Console.WriteLine(BitConverter.ToString(snmp_packet));
            Console.WriteLine(snmp_packet.Length);
            return snmp_packet;
        }
        public byte[] convertMessage(String[] mibValues)
        {
            byte[] temp = new byte[32];
            int value;
            int counter = 0;
            for (int i = 0; i < mibValues.Length; i++)
            {
                value = Convert.ToInt16(mibValues[i]);
                if(value > 127)
                {
                    temp[counter++] = Convert.ToByte(128 + (value / 128));
                    temp[counter++] = Convert.ToByte(value - ((value/128)*128));
                    mMibLength++;
                }
                else
                temp[counter++] = Convert.ToByte(mibValues[i]);
            }
            return temp;
        }
    }
}
```

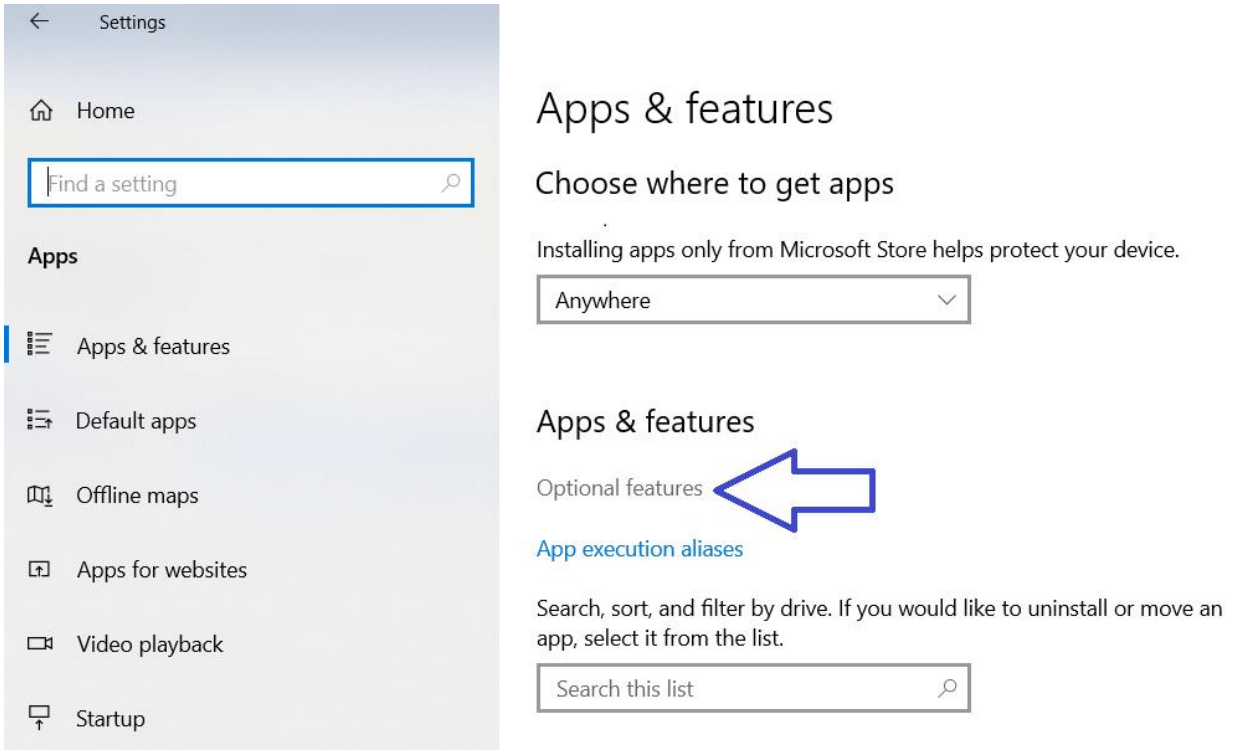
Appendix (C)

Install SNMP and WMI in Windows:

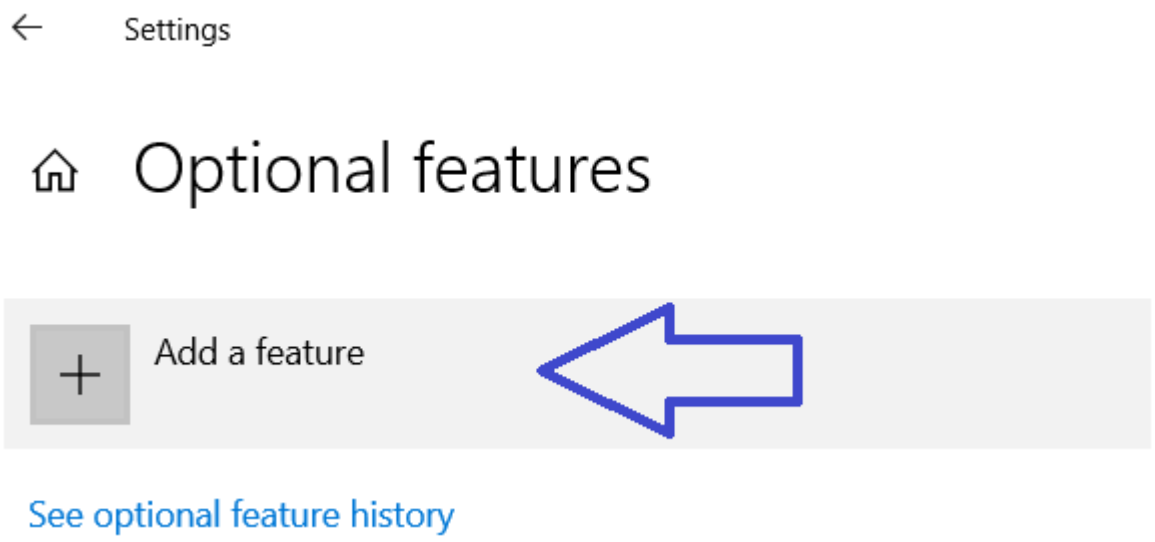
In the beginning, if the system is not Windows Server, go to the settings as shown in the figure.



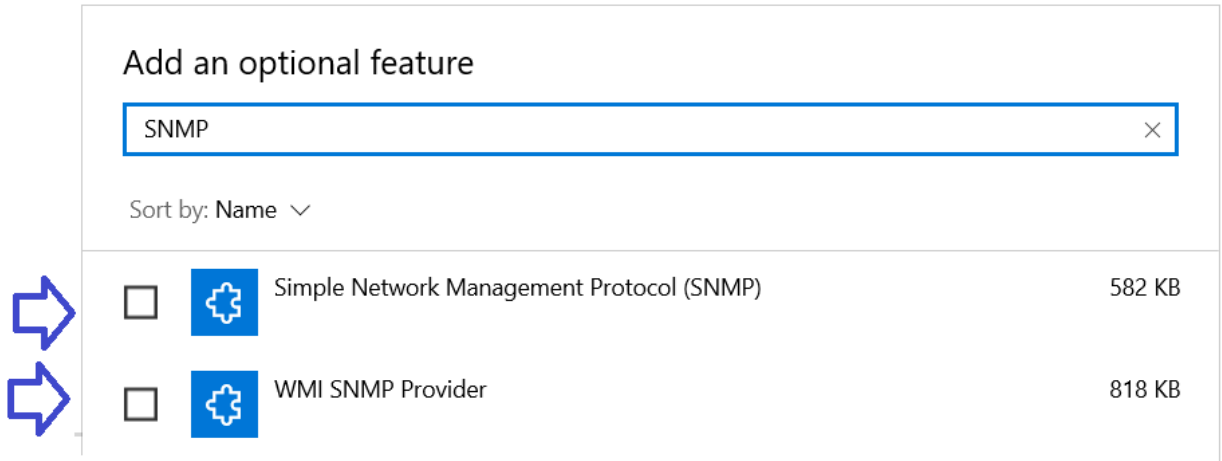
Choose the optional features.



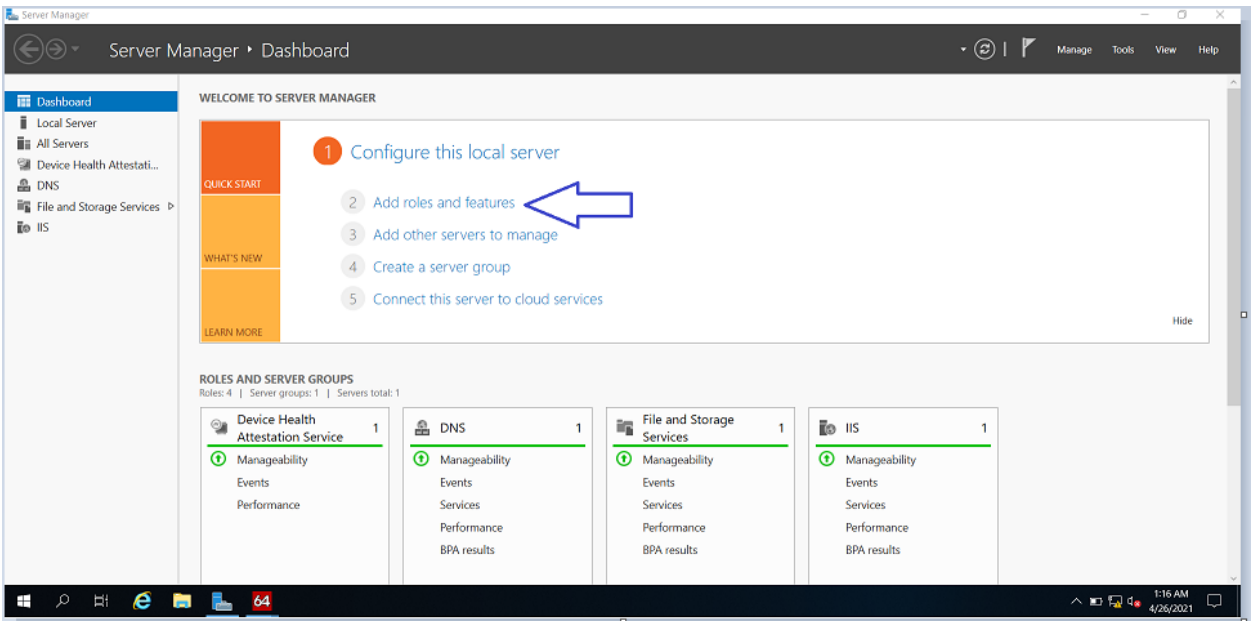
Select Add a feature.



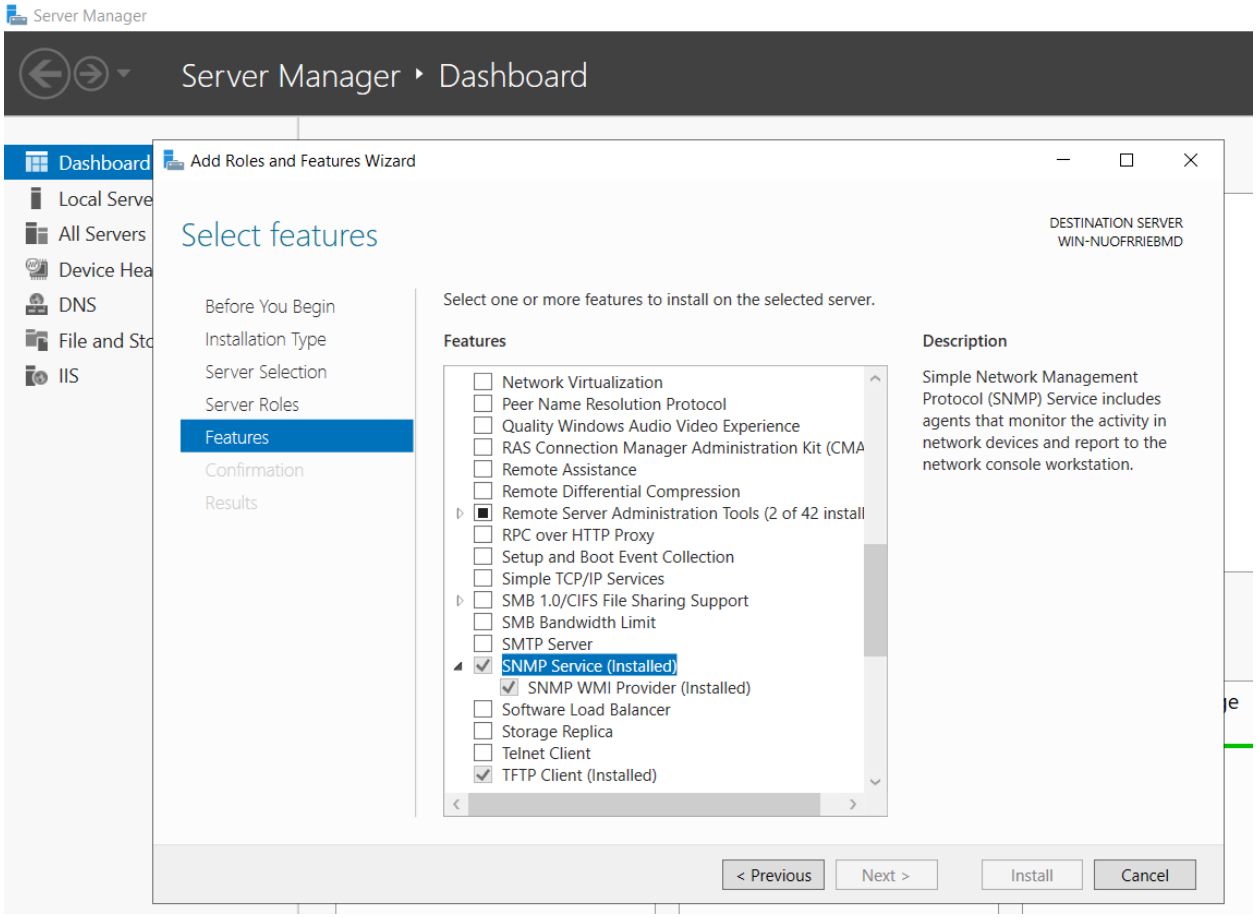
Select all and download



If you are using Windows Server, activation is through server manager, then chose to add roles and features.

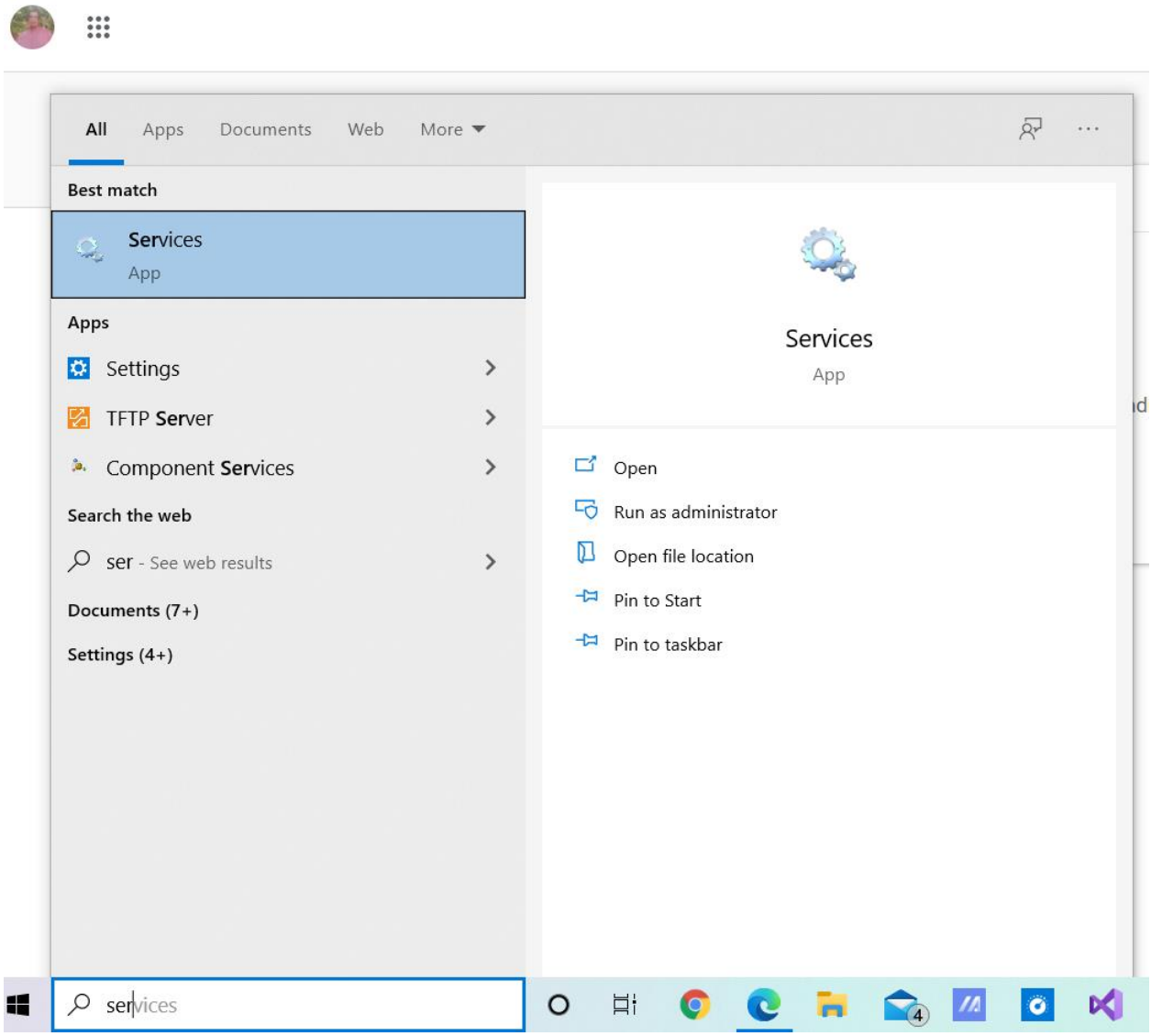


Add it if it is not installed in the system.

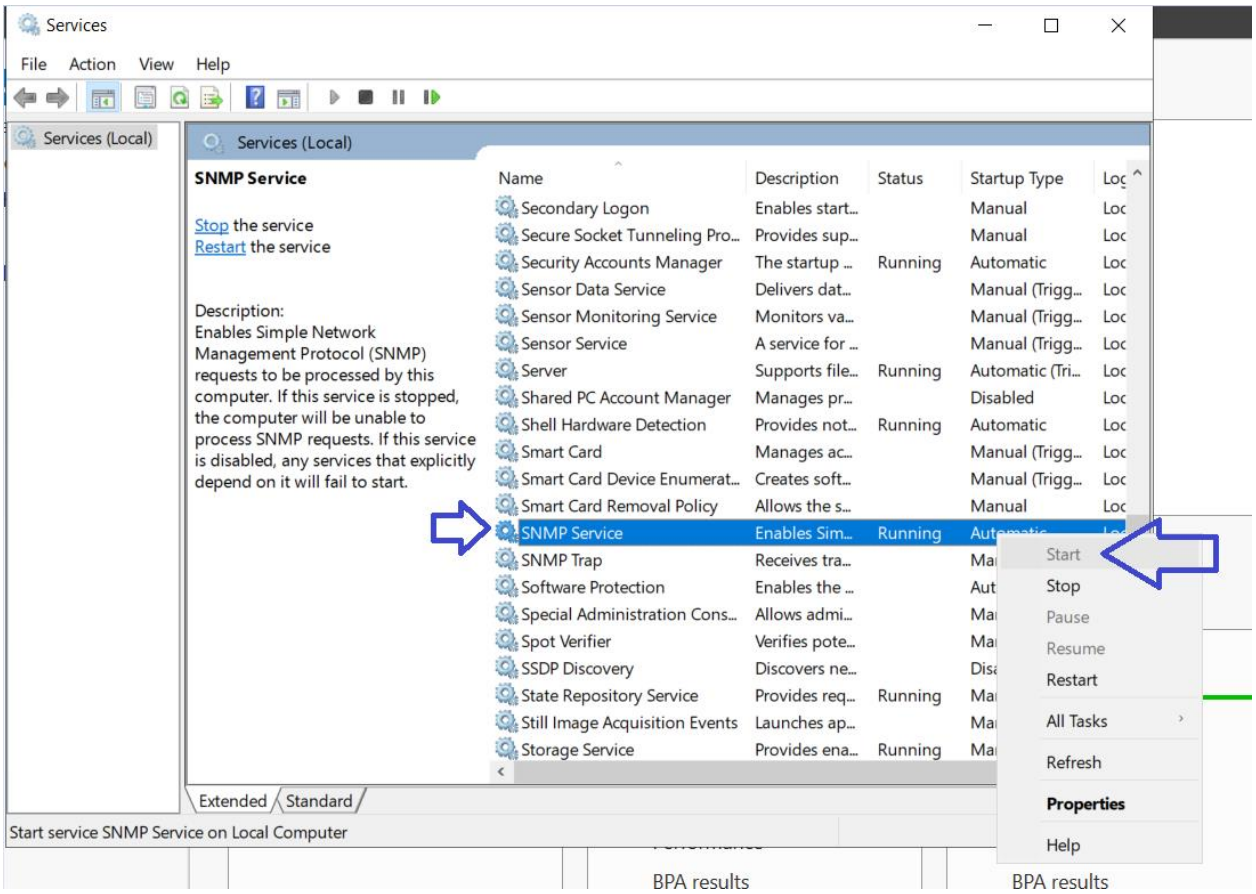


Enable SNMP in Windows:

Open the services by search by windows logo.



Select to start an SNMP services.

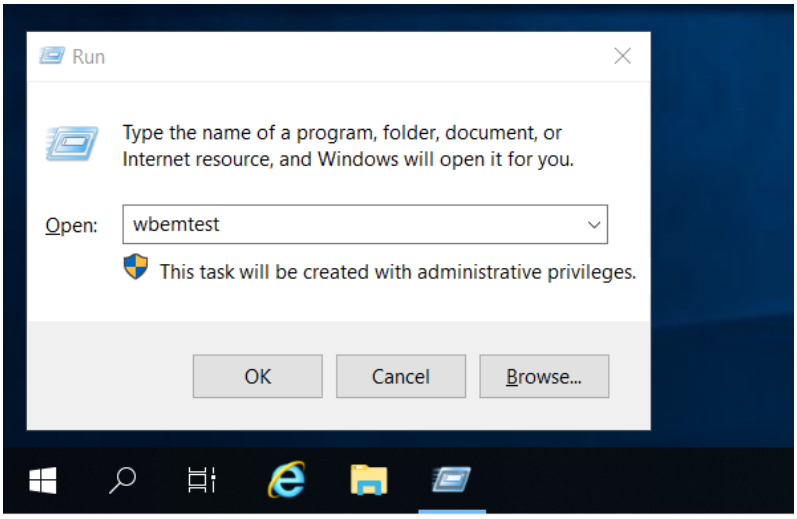


Finally, from the security tab add a community with the name of the public for example. Also, select an option to accept packets from any host.

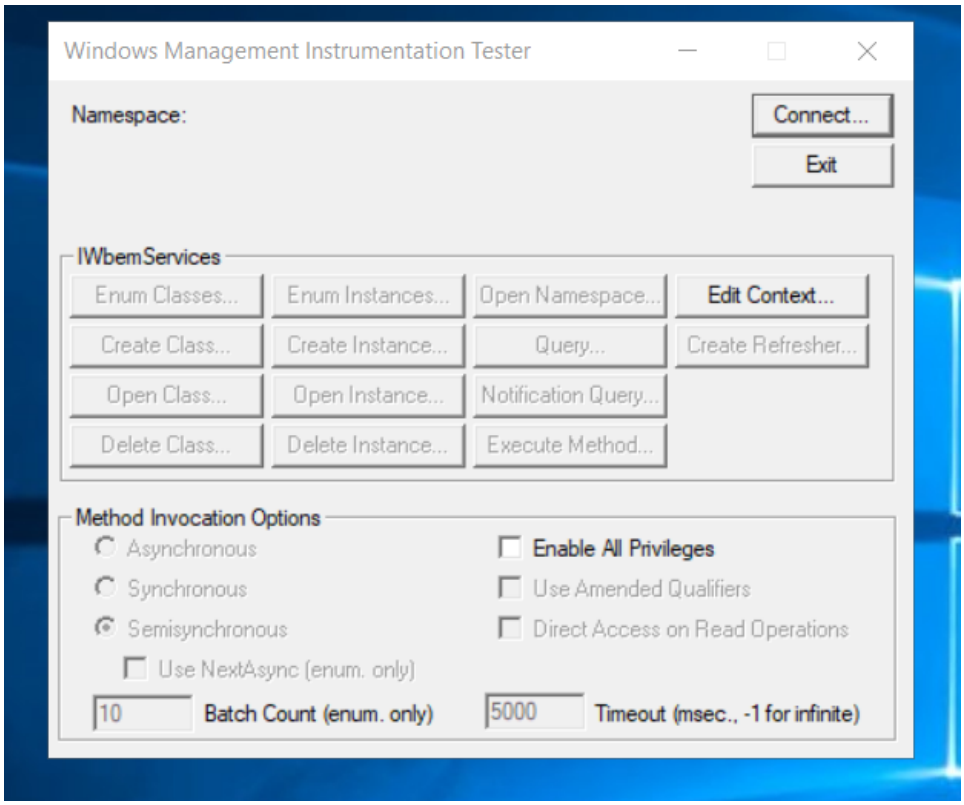
Appendix (D)

A utility for a tester for WMI:

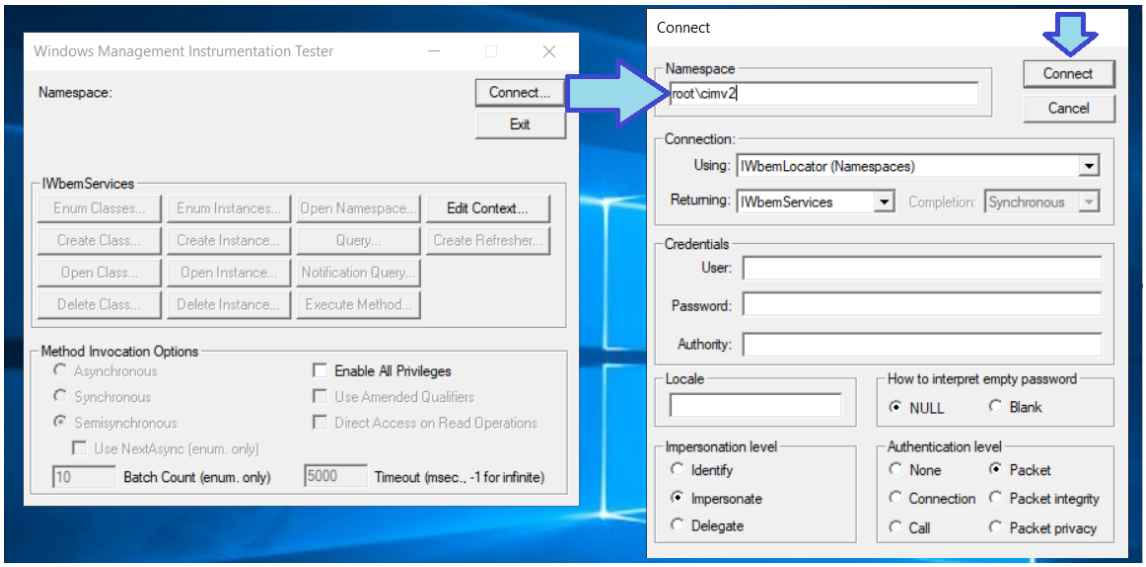
Sometimes we want to try a specific service before we apply it via code. We can enter by pressing the Windows button with the letter R to write [wbemtest].



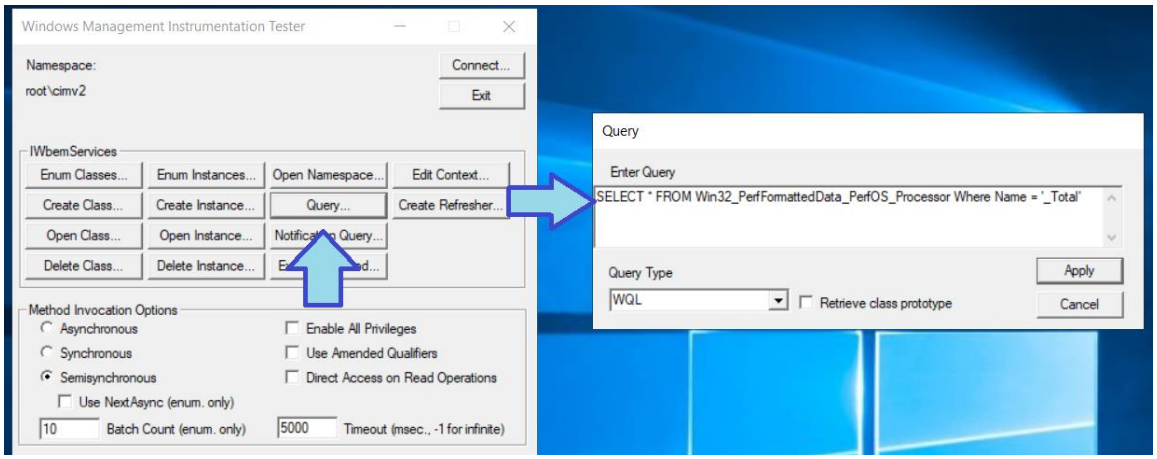
This program will appear with various features. You can start using it with a button connect.



And try it according to the service where you are, and press connect.



Now we can enter a specific query and try to find out, for example, the total CPU consumption value, which is what we use in the code



To be accurate, the experience has done the simulation of the processor consumption to reach a hundred through the AIDA64 program, and we note that a list of all the details of the query is shown to us, and indeed it was revealed to us that it reached 100%