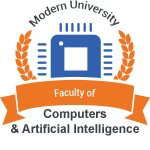
****



**Network Monitoring System**

**Graduation Project 2022**

**Submitted by**

**1-Hossam Gamal Mohamed**

**2-Abdelrahman Elsayed Mohamed**

**3-Beshoy Sameh Eldabaa**

**4-Hassan Hesham Mohamed**

**5-Ammar Yasser Mohamed**

**6-Youssef Elsayed Mostafa**

**Supervised by**

**Instructor: Dr.Assem Al Ashker**

**T.A: Hazem Omar**

**T.A: Omar Khaled**

**Acknowledgment**

**Our greatest gratitude to Modern University for Technology and Information (M.T.I) for the great role it played in forming our character and personalities, providing us with all means necessary for our educational development and providing us as well with a wonderful environment to learn in throughout our educational years.**

**This would not have been possible unless Prof. Dr. Olfat Kamel, President of the M.T.I University has established such a learning environment, providing the most effective facilities for our job to be done.**

**It is a pleasure to thank those who maneuvered this Possible, Prof. Dr.Mohamed Taher ElMayah Dean of the faculty of Computers and Information & Artificial intelligence for his unlimited help, support, encouragement and guidance.**

**A Special thanks to our head of the Computer Science Department Prof. Dr. Hanafy Isamil, we are Honored and got the pleasure of working with as our supervisor.**

**A Special thanks to Dr. Assem Al Ashkar , we are Honored and got the pleasure of working with as our supervisor and appreciate your efforts with us while the year and graduation project.**

**Special thanks to all Mti Family.**

**Special thanks to our families who always supported us during our academic life, and we could not do it without them with us.**

**Table of Content**

[**Chapter One Introduction** 8](#_Toc135627451)

[**1.1** **Basic Concepts** 8](#_Toc135627452)

[1.1.1 Network Monitoring 8](#_Toc135627453)

[1.1.2 Performance monitoring 9](#_Toc135627454)

[1.1.3 Network Throughputs 9](#_Toc135627455)

[1.1.4 System Throughputs 9](#_Toc135627456)

[1.1.5 Network Performance Analysis 9](#_Toc135627457)

[1.1.6 Path ways 9](#_Toc135627458)

[**1.2 Importance & Motivations** 10](#_Toc135627459)

[1.2.1 Importance of Network Monitoring Applications 10](#_Toc135627460)

[1.2.2 Importance of Our project 11](#_Toc135627461)

[**1.3 Problem Definition** 11](#_Toc135627462)

[1.3.1 Objectives 11](#_Toc135627463)

[**1.4 Solution Approach** 12](#_Toc135627464)

[**1.5 Documentation Organization** 12](#_Toc135627465)

[**Chapter Two Project Planning** 15](#_Toc135627466)

[**2.1 Project Tasks** 15](#_Toc135627467)

[**2.2 Project Management:** 18](#_Toc135627468)

[**2.3 Software Tool for planning:** 20](#_Toc135627469)

[2.3.1 Project Subsystems 21](#_Toc135627470)

[2.3.2 Gantt Chart 21](#_Toc135627471)

[**Chapter Three Alternative Systems** 24](#_Toc135627472)

[3.1 Auvik 24](#_Toc135627473)

[3.1.1 Features of Auvik: 24](#_Toc135627474)

[3.1.2 Advantages of Auvik: 24](#_Toc135627475)

[3.1.3 Disadvantages of Auvik: 24](#_Toc135627476)

[3.1.4 Screenshots for Auvik: 25](#_Toc135627477)

[**3.2 PRTG** 26](#_Toc135627478)

[3.2.1 Features of PRTG: 26](#_Toc135627479)

[3.2.2 Advantages of Auvik: 26](#_Toc135627480)

[3.2.3 Disadvantages of Auvik: 26](#_Toc135627481)

[3.2.4 Screenshots for PRTG: 27](#_Toc135627482)

[**3.3 Domotz** 28](#_Toc135627483)

[3.3.1 Features of Domotz: 28](#_Toc135627484)

[3.3.2 Advantages of Domotz: 28](#_Toc135627485)

[3.3.3 Disadvantages of Domotz : 28](#_Toc135627486)

[3.3.4 Screenshots for Domtoz: 29](#_Toc135627487)

[**Chapter Four System Analysis & Design** 32](#_Toc135627488)

[**4.1 Functional view** 32](#_Toc135627489)

[4.1.1 Maintain basic data 32](#_Toc135627490)

[4.1.2 Transactions for the system: 32](#_Toc135627491)

[**4.2** **Use case diagram** 34](#_Toc135627492)

[4.2.1 Use case diagram 34](#_Toc135627493)

[**4.3** **Database design using Class diagram** 35](#_Toc135627494)

[4.3.1 Class Diagram 35](#_Toc135627495)

[4.4 Sequence Diagram 36](#_Toc135627496)

[4.4.1 Sequence Diagram 36](#_Toc135627497)

[**4.5** **Activity Diagram** 37](#_Toc135627498)

[4.5.1 Activity Diagram 37](#_Toc135627499)

[**4.6** **Block Diagram** 38](#_Toc135627500)

[4.6.1 Block Diagram 38](#_Toc135627501)

[**4.7** **Database Tables** 39](#_Toc135627502)

[**Chapter Five System Implementation** 41](#_Toc135627503)

[**5.1 software & hardware used** 41](#_Toc135627504)

[5.1.1 Visual Studio Code 41](#_Toc135627505)

[5.1.2 QT Designer 42](#_Toc135627506)

[5.1.2 Python 42](#_Toc135627507)

[**5.3 Graphical User Interface (GUI)** 44](#_Toc135627508)

[**Chapter Six System Testing** 50](#_Toc135627509)

[**6.1 Testing Importance** 50](#_Toc135627510)

[**6.2 System Testing Evaluation** 51](#_Toc135627511)

[**6.3 Tests we performed** 52](#_Toc135627512)

[6.3.1 Unit testing 52](#_Toc135627513)

[6.3.2 Integration testing 52](#_Toc135627514)

[6.3.3 Test case scenario: 52](#_Toc135627515)

[**Chapter Seven Conclusion And Future Work** 59](#_Toc135627516)

[**7.1 Conclusion** 59](#_Toc135627517)

[**7.2 Future Work** 59](#_Toc135627518)

[**Chapter Eight References** 61](#_Toc135627519)

[**Chapter Nine Appendix** 63](#_Toc135627520)

[**9.1 Program.py** 63](#_Toc135627521)

**Summary**

A Python program designed to monitor networks can be used to analyze and troubleshoot network issues. The program would use various network monitoring techniques, such as pinging network devices, to collect data and analyze network performance.

The program would typically use the Simple Network Management Protocol (SNMP) to gather information about network devices and their configurations. It may also use the Internet Control Message Protocol (ICMP) to send ping requests and receive responses from network devices.

The program can be designed to monitor various aspects of a network, such as network bandwidth, latency, packet loss, and uptime. It can also analyze network traffic patterns and identify potential security threats.

In addition to monitoring network performance, the program can generate alerts when it detects issues or anomalies on the network. These alerts can be sent to network administrators via email, text message, or other notification methods.

network system analysis is critical for maintaining the stability, performance, and security of a computer network. By identifying issues and optimizing network performance, network administrators can ensure that their networks are running smoothly and efficiently, which is essential for supporting business operations and user productivity.

Overall, a Python program designed to monitor networks can be a valuable tool for network administrators to ensure the optimal performance and security of their networks.

**CHAPTER ONE**

**Introduction**

# **Chapter One Introduction**

**This chapter introduces the background of our project.**

1.1 describes the basic concepts of the project.

1.2 describes importance of this project.

1.3 shows the main project problem definition, objective.

1.4 discuss the solution approach of the project.

1.5 discuss the documentation organization of the project.

## **Basic Concepts**

**This part explains the basic concepts related to the project.**

**In This part we explain the basic concepts related to the project.**

• discusses Network Monitoring the title of our project

• will also discusses concepts related to Network Monitoring like Network Throughputs, Network Performance Analysis, Communication Lines, Transmission Lines, Data Handling.

### Network Monitoring

Network monitoring is the process of constantly monitoring a computer network for problems such as slow traffic or component failure and detect connected devices.

Network Monitoring tools are always scanning the network and are designed to automatically notify network administrators via text.

Benefits of network monitoring

The most important benefit of network monitoring is visibility, easy to understand, at a glance picture of all connected devices across the organization, and visualizing the flow of data across devices and networks enables the network administrators to quickly identify and remediate any problems which should be solved quickly as possible as they can.

### Performance monitoring

Network performance refers to measures of service quality of a network as seen by the customer.

is a systematic and periodic observation of performance over time to develop or verify performance records, uncover inefficient and ineffective practices, identify needs for services, and most importantly, detect underperformance timely to avoid further deterioration of performance.

### Network Throughputs

refers to the rate of successful message delivery over a communication channel, such as Ethernet or packet radio, in a communication channel. The data that these messages contain may be delivered over physical or logical links, or through network nodes. Throughput is usually measured in bits per second and sometimes in data packets per second or data packets per time slot.

### System Throughputs

The sum of the data rates that are delivered to all terminals in a network. The throughput of a communication system may be affected by various factors, including the limitations of the underlying analogue physical medium, available processing power of the system components, end-user behavior, etc.

### Network Performance Analysis

Network Performance Monitoring and Analysis delivers a historical and real-time view of how well the network delivers services and applications. It delivers metrics that matter, including user quality of experience, scores with detailed transport measurements for application-aware performance information, which is critical for identifying trends impacting network experience

### Path ways

• communication lines used for information systems at government agencies.

•Transmission Lines Used for the transmission of electrical power from generating substation to the different distribution units.Difference Between Communications Lines and Transmission Lines:

- Communication means 2-way talking, such as in a network where computers talk to each other to transfer data from one computer to another. Communication can also mean one station talking to many stations at once, or you can have many stations talking to many stations at once.

- Transmission is only one way of communication, which is the computer can only talk out and not receive anything. The transmission would be half of the communication where you would send out a packet to the network and then the other half of the communication would use the receiver to accept the packet as it returns.

## **1.2 Importance & Motivations**

**Our application is necessary and important for Network Security. Our project is to Design and develop a stimulation of the basic concepts of Network Monitoring System. This part included the importance of project and how helped the network to be secured and tracked.**

## 1.2.1 Importance of Network Monitoring Applications

Network monitoring applications are essential tools for managing modern computer networks. Here are some of the key reasons why network monitoring applications are important:

Identifying network issues: Network monitoring applications can detect issues such as network outages, latency, packet loss, bandwidth issues, and other problems that can impact network performance.

Troubleshooting network problems: Network monitoring applications can help network administrators identify the root cause of network issues, speeding up the troubleshooting process and reducing downtime.

Ensuring network security: Network monitoring applications can detect security threats such as malware, viruses, and hacking attempts. They can also monitor network traffic patterns to identify suspicious activity and prevent security breaches.

Optimizing network performance: Network monitoring applications can collect data on network usage, bandwidth, and other metrics, allowing network administrators to optimize network performance, identify potential bottlenecks, and plan for future capacity needs.

Meeting regulatory compliance: Many industries are subject to regulatory compliance requirements, such as HIPAA, PCI-DSS, and GDPR. Network monitoring applications can help organizations meet these requirements by ensuring that sensitive data is being transmitted securely and that networks are being monitored for security threats.

Improving customer experience: Network monitoring applications can help organizations ensure that their networks are performing optimally, which can have a positive impact on customer experience and satisfaction.

## 1.2.2 Importance of Our project

• **delivers a historical and real-time** view of how well the network is delivering services and applications

**• detect devices and other elements** that comprise or touch the network, as well as provide status updates.

**• control network traffic** through managing or prioritizing data and can monitor network traffic by measuring the amount and types of traffic

**• allows administrators and IT** teams to react quickly to any network issues.

## **1.3 Problem Definition**

* indicates a device is offline when it is not.
* weak UI that can’t be addressed.
* gets updated very regularly, which caused interruption tracking of networks issues with sensors and tracking
* can’t provide reports or issues exists

### 1.3.1 Objectives

**The main objective of this project is to develop an application for a network monitoring system, that allows administrators and IT teams to react quickly to any network issues. It tracks all details about Network as usage, bandwidth, etc.**

**Our project introduces a Network Monitoring system with the objective to analyze the network and monitoring everything on network and solve issues or failures.**

## **1.4 Solution Approach**

* **Provide perfect tracking**
* **Simple and good UI Design**
* **Updates in specific dates**
* **Provide Reports for network tracking**

## **1.5 Documentation Organization**

**Chapter 1: (introduction)**

Introduction which includes a brief description of the different basic concepts related to the project that includes the Network Monitoring system and the Importance of Network Monitoring Applications. And the importance of our application, problem definition, objectives and also the solution approach.

**Chapter 2: (System Planning)**

Project planning includes project team tasks and distribution among the project group members.

**Chapter 3: (Alternative Systems)**

Alternative Systems shows some examples of related systems with screenshots and show the advantages and disadvantages of these systems and also a comparison that shows the pros and cons of the systems.

**Chapter 4: (System analysis and design)**

System analysis and design show the points of how the user will use the system. The use case diagram is going to show the functionalities of the system and the operations made in the system. The application interaction diagrams are needed to design the overall application attributes and methods and the interaction between different objects.

**Chapter 5: (System Implementation)**

System implementation highlights the application development tools. There is a detailed descriptive screenshot step-by-step for how we used these tools. the application’s screen shots and user interface are shown and described as well.

**Chapter 6: (System Testing)**

This Chapter discusses the testing and evaluation of all previous work, the testing process shows that the system has been tested during and after finishing the system implementation. The evaluation of the system was achieved by the users who actually Tested it.

we gathered real people to testing on our program!

**Chapter 7: (Conclusion and future work)**

Includes the conclusion of the implemented application and the future work that can be per-formed to the project.

**CHAPTER TWO**

**Project Planning**

## **Chapter Two Project Planning**

Project planning is the process of defining project goals, objectives, timelines, and resources required to achieve those goals. It involves creating a detailed roadmap or blueprint for a project, outlining the tasks to be completed, the resources needed, and the deadlines for each task.

## **2.1 Project Tasks**

**The project includes the following tasks:**

1. Problem Definition
2. Collect Data
3. System Analysis & Design

* Block Diagram
* Use Case Diagram
* Class Diagram
* Sequence Diagram
* Activity Diagram
* Database Tables
* System Design

1. System Implementation

* Database
* Programming
* Performance
* Comparison
* Smart GUI Module
* Integration

1. System Testing
2. System Documentation
3. System Presentation
4. System Demo

**The description of the Tasks:**

1. **Problem Definition:** Identify a clear problem that is the focal point of our project that the project will address and improve. Where defining the problem is the first step towards a successful project.
2. **Collect Data:** the collecting of information about related system of our project and knowing how they work, their advantages and disadvantages and compare these systems. to configure features of our project.
3. **System Analysis & Design:** the process of analyzing and designing information systems to meet the specific needs of an organization. This process involves understanding the current system, identifying the problems or areas for improvement, and designing a new system that addresses those issues.

* **Block Diagram:** is a diagram that represents the components of a system or process using blocks connected by lines or arrows. It is a visual representation of a system or process that shows the relationships between its various components.
* **Use case Diagram:** that represents the interactions between actors (users or external systems) and a system or application. It is used to define, visualize, and communicate the functional requirements of a system or application.
* **Class Diagram:** represents the classes, interfaces, and their relationships to each other in a system or application. It is used to model the static structure of a system or application, defining the classes, their attributes, and methods, and the relationships between them.
* **Sequence Diagram:** shows the interactions between objects or components in a system or application over time. It is used to model the dynamic behavior of a system, showing how objects interact with each other to accomplish a task or goal.
* **Activity Diagram:** models the flow of activities or actions in a system or application. It is used to model the dynamic behavior of a system, showing how objects interact with each other to accomplish a task or goal.
* **Database Tables:** the fundamental building blocks of a relational database. They are used to store and organize data in a structured way, making it easier to manage and retrieve the data when needed.
* **System Design:** the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specific requirements. It involves transforming the requirements into a detailed specification that can be implemented and tested.

**4-System Implementation:** This involves building and testing the new system and ensuring that it meets the requirements identified during system analysis & design.

* **Database:** structured collection of data that is stored and organized in a way that allows for efficient retrieval and manipulation of the data. Databases are used to store and manage vast amounts of data for a wide variety of applications, including business, science, education, and government.
* **Programming:** the process of writing and testing the actual code that will be used to build a software system or application. This stage comes after the design and planning stages, where the requirements and specifications of the software have been defined & the architecture & modules have been designed.
* **Performance:** refers to the ability of a software system or application to meet performance requirements under normal and peak loads. Performance is a critical aspect of software development as it directly impacts the user experience and the success of the system or application.
* **Comparison:** involves evaluating different approaches, technologies, or solutions to determine which one is the best fit for the software system or application being developed. Comparison can help ensure that the resulting software meets the requirements, is efficient, and scalable, and provides a good user experience.
* **Smart Gui Module:** involve designing and implementing the GUI module.

One approach to implementing a GUI module is by using a smart GUI framework or library.

* **Integration:** the process of combining different modules, components, or systems into a single, functioning software system or application. Integration is a critical part of the software development process, as it ensures that all the individual pieces of the software work together seamlessly to meet the requirements of the project.

1. **System Testing:** Is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computerbased system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is a series of different tests whose sole purpose is to exercise the full computer-based system.
2. **System Documentation:** The collection of documents that describes the requirements, capabilities, limitations, design, operation, and maintenance of a system, such as communications, computing, or information processing system.
3. **System Presentation:** Representation of the final work and everything that was done in the project by explaining what was learned and what was done by whom and how long it takes.
4. **System Demo:** shows how a software system or application works. The purpose of a system demo is to give a clear understanding of the software system or application and its capabilities.

**2.2 Project Management:** the process of planning, organizing, and overseeing the execution of a project from start to finish. It involves the application of knowledge, skills, tools, and techniques to meet project requirements and goals while adhering to scope, schedule, and budget constraints.

**Project Tasks:**

the specific activities or actions that need to be completed in order to achieve the goals and objectives of a software development project. Project tasks typically include activities such as planning, design, implementation, testing, deployment, and maintenance.

|  |  |  |
| --- | --- | --- |
| **Task #** | **Task Name** | **Student Name** |
| 1 | Problem Definition | All Students |
| 2 | Collect Data | All Students |
| 3 | System Analysis | All Students |
| 4 | System Design | All Students |
| 5 | Simple Database Module | Ammar,Beshoy,Hassan |
| 6 | Programming Activities | Hossam,Ammar,Beshoy |
| 7 | Performance Activities | Ammar,Beshoy,Abdelrahman |
| 8 | Comparison Activities | Abdelrahman,Hassan,Youssef |
| 9 | Smart Gui Module | Hassan,Youssef |
| 10 | Integration | Hossam,Ammar |
| 11 | System Testing | Hossam, Abdelrahman,Youssef |
| 12 | System Documentation | All Students |
| 13 | System Presentation | Beshoy,Abdelrahman,Hassan |
| 14 | System Demo | Hossam,Ammar,Beshoy |

**Tasks Duration:**

the amount of time required to complete specific tasks in a software development project. Accurately estimating task duration is essential to project planning and helps ensure the project stays on schedule.

|  |  |  |
| --- | --- | --- |
| **Task #** | **Duration (weeks)** | **Student Name** |
| 1 | 1 | All Students |
| 2 | 1 | All Students |
| 3 | 4 | All Students |
| 4 | 4 | All Students |
| 5 | 2 | Ammar,Beshoy,Hassan |
| 6 | 2 | Hossam,Ammar,Beshoy |
| 7 | 2 | Ammar,Beshoy,Abdelrahman |
| 8 | 2 | Abdelrahman,Hassan,Youssef |
| 9 | 2 | Hassan,Youssef |
| 10 | 2 | Hossam,Ammar |
| 11 | 4 | Hossam, Abdelrahman,Youssef |
| 12 | 4 | All Students |
| 13 | 4 | Beshoy,Abdelrahman,Hassan |
| 14 | 4 | Hossam,Ammar,Beshoy |

## **2.3 Software Tool for planning:**

**Task made by (Microsoft project planning version 2016)**

**Microsoft Project** is a project management software tool that provides a range of features and functionality to help manage and track project tasks, timelines, resources, and budgets. Some of the key abilities of Microsoft Project include:

Project planning and scheduling: Microsoft Project allows users to create project plans and schedules, defining tasks, durations, dependencies, and resource requirements. Users can also create Gantt charts and other visual representations of the project schedule.

Resource management: Microsoft Project allows users to manage project resources, including personnel, equipment, and materials. Users can assign resources to specific tasks, monitor resource availability and utilization, and adjust resource allocations as needed.

Budget management: Microsoft Project allows users to manage project budgets, including tracking costs and expenses, creating budget reports, and monitoring budget variances.

Task management: Microsoft Project allows users to manage project tasks, including creating and assigning tasks, tracking task progress, and identifying and addressing task issues and delays.

Collaboration and communication: Microsoft Project allows users to collaborate and communicate with team members, stakeholders, and other project participants using features such as email integration, task comments, and team member status updates.

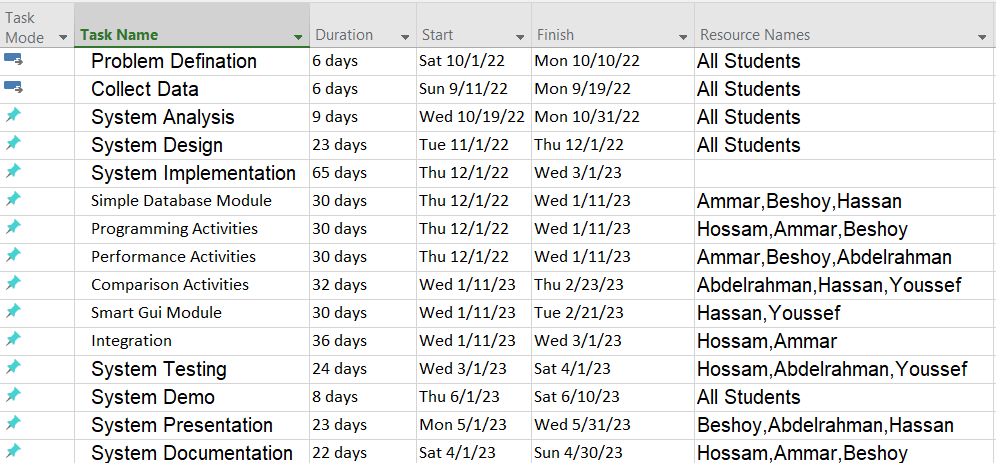
Reporting and analysis: Microsoft Project allows users to create a range of reports and analyses, including project status reports, resource utilization reports, and budget reports. Users can also use built-in analytics tools to analyze project data and identify trends and patterns.

### 2.3.1 Project Subsystems

Group of interconnected and interactive parts that performs an important job or task as

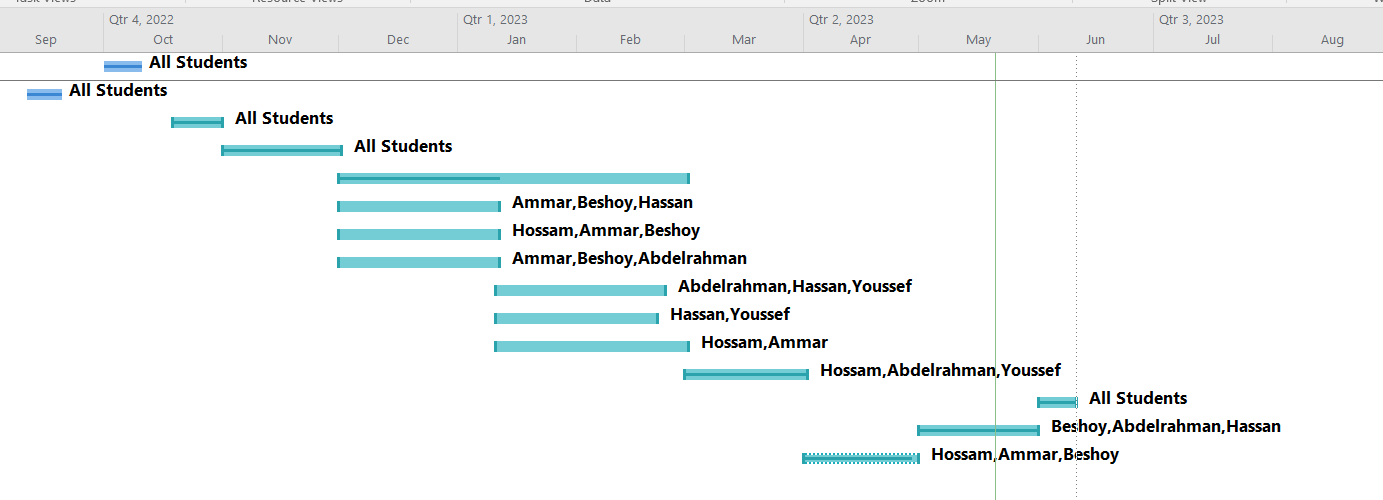
a component of a larger system. A subsystem, while a system in itself, is also wholly

contained within a larger system.



### 2.3.2 Gantt Chart

commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.



**Conclusion**

In any software development project, planning is a critical component that lays the foundation for the project's success. The planning phase helps to define the project's goals, objectives, and requirements, as well as identify the resources needed to complete the project. Effective planning can help ensure that the project stays on track, is completed on time, within budget, and to the required quality standards.

During the planning phase, project managers and team members typically engage in a range of activities, including defining project scope, gathering requirements, creating a project plan and schedule, identifying risks, and allocating resources. Effective planning involves careful consideration of various factors, including project goals and objectives, stakeholder needs and expectations, resource availability, and potential risks and challenges.

A well-planned project typically includes a clear project plan and schedule, a well-defined scope and requirements, realistic budget and resource allocations, and a risk management plan. Effective communication and collaboration among project stakeholders are also essential for successful planning and execution of the project.

In conclusion, effective planning is critical for the success of any software development project. By carefully planning and preparing for a project, project managers and team members can help ensure that the resulting software system or application meets the needs and expectations of its intended users and stakeholders and is delivered on time, within budget, and to the required quality standards.

**CHAPTER THREE**

**Alternative Systems**

## **Chapter Three Alternative Systems**

Alternative Systems shows some examples of related system some examples of GUI to the advantage and disadvantages of these system and comparison that shows the pros and cons of the systems.

3.1 Auvik 

cloud-based network management software gives users insight into the networks they manage, and automates complex and time-consuming network.

### 3.1.1 Features of Auvik:

* Automate network visibility & IT asset management Complete your network picture with automated network discovery, inventory, and documentation that updates in real-time as the network evolves.
* Simplify network performance monitoring & troubleshooting Respond to network issues in real-time and dive deep into problems with Syslog so your users are always connected to the business critical resources they need to do their job.
* Automate configuration backup & recovery Sleep easy knowing you have access to up-to-date device configurations, a snapshot of all historical configs, and the ability to export or restore a configuration with ease.
* Intelligently analyze network traffic Go Beyond Net Flow to easily identify who’s on the network, what they’re doing, and where their traffic is going with Auvik TrafficInsights.

### 3.1.2 Advantages of Auvik:

* Auvik takes minutes to install and automatically begins discovering the entire network. Combined with its intuitive design. Intelligently analyze network traffic.
* Remotely connect network devices in Auvik’s inventory and troubleshoot issues without ever leaving your desk.
* Dynamic topology map
* Automated inventory and documentation
* Configuration backups

### 3.1.3 Disadvantages of Auvik:

* The map is not easy to quickly zoom in and out.
* weak UI that has never been addressed.

### 3.1.4 Screenshots for Auvik:

**Graphical user interface, chart

Description automatically generated**

**Chart, line chart

Description automatically generated**

****

## **3.2 PRTG**

flagship offering from German software company Paessler, for monitoring local and wide area networks (LANs & WANs), servers, websites, apps, and more.

### 3.2.1 Features of PRTG:

* Network analysis modules to automatically find network devices and sensors.
* Distributed monitoring to monitor several networks in different locations.
* Customizable alerts for specific needs: Various notification methods: email, push, SMS text messages, Syslog messages and Simple Network Management Protocol (SNMP) traps, HTTP requests, event logs, Amazon Simple Notification Service (SNS), executing scripts. Multiple ways to trigger notifications: status alerts, limit alerts, threshold alerts, multiple condition alerts, escalation alerts.

### 3.2.2 Advantages of Auvik:

* High security standards: Secure Sockets Layer (SSL)/Transport Layer Security (TLS) secured connections and web servers, secure ciphers, personalized user rights management, and much more.
* Low system requirements: To run PRTG Network Monitor, an average PC that is not older than 2 years is enough. Even a netbook can monitor more than a thousand sensors. PRTG Hosted Monitor requires no hardware for the PRTG core server.
* High performance: The database system stores raw monitoring results as well as logs, Toplists, and tickets. This outperforms Structured Query Language (SQL) servers for monitoring data. You can distribute high loads among several probes and also access the database through the PRTG API.
* Graphics engine for user-friendly live graphs and historic data graphs.

### 3.2.3 Disadvantages of Auvik:

* The software gets updated very regularly. Whilst this is usually a good thing to fix bugs etc, it does meantime downtime of the monitoring quite often while they are installed.

### 3.2.4 Screenshots for PRTG:

**Graphical user interface, application, table, Excel

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

## **3.3 Domotz**



Domotz offers network monitoring and management software for MSPs, Integrators, Security Professionals, and Business Owners. The platform has integrated features such as remote device access, SNMP monitoring, network diagnostics, remote power management, device alerts, team collaboration, network mapping, and multiple others.

### 3.3.1 Features of Domotz:

* Monitoring Identify issues and troubleshoot in real time at network and device level.
* Management Use Domotz to connect remotely and resolve issues just like you would if you were on the customer site.
* Security Automatically scans networks for security vulnerabilities with network security scans.
* Inventory Gain full knowledge of the networks you manage with real-time automated devices and attributes discovery.

### 3.3.2 Advantages of Domotz:

* The ability to see what is on a client's network, whether we manage it or not, is very helpful. Also being able to access those devices without remote access to a PC on site makes life so much easier.
* Easy setup with multiple/flexible options.
* Responsive Navigation, Interface .

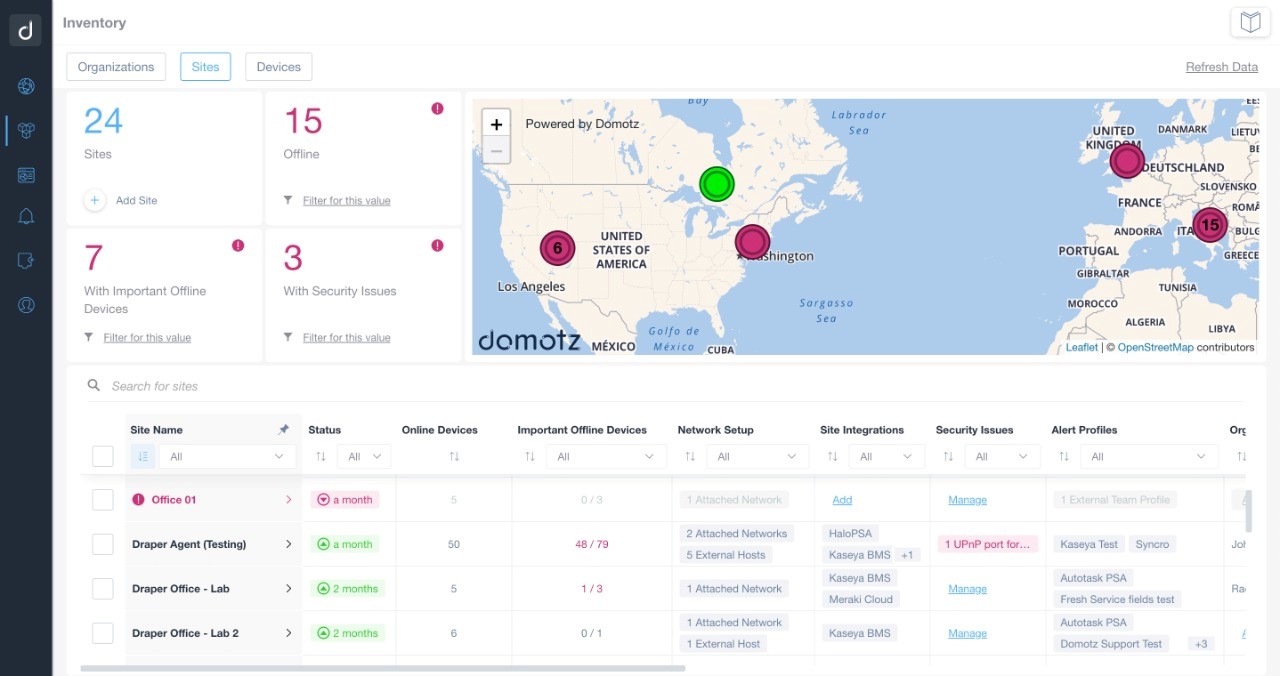
### 3.3.3 Disadvantages of Domotz :

* Sometimes indicates a device is offline when it is not.

### 3.3.4 Screenshots for Domtoz:

Diagram, engineering drawing

Description automatically generated



**Conclusion**

Alternative System in system refer to research and studies that are related to the development of a software system or application. These works can include academic research, industry reports, case studies, and other publications that provide insights and best practices related to the development, implementation, and maintenance of software systems.

The review of related works is an important part of the software development process, as it helps developers gain a better understanding of the current state of the field, and identify best practices and lessons learned from previous projects. By drawing on the insights and experiences of other researchers and practitioners, developers can help ensure that their software system or application is well-designed, meets the needs of its intended users and stakeholders, and is delivered on time and within budget.

Some of the key findings from related works in system development include the importance of effective project management, the value of user-centered design, the benefits of agile methodologies, and the importance of quality assurance and testing. Other related works have highlighted the importance of collaboration and communication among project stakeholders, the need for ongoing maintenance and updates to software systems, and the challenges of managing risks and uncertainties in software development projects.

In conclusion, the review of related works is an essential part of the software development process, as it helps developers gain a deeper understanding of the field, identify best practices and lessons learned from previous projects, and inform the design and implementation of new software systems or applications. By drawing on the insights and experiences of other researchers and practitioners, developers can help ensure that their software systems orapplications are well-designed, meet the needs of their intended users and stakeholders, and are delivered on time and within budget. Ultimately, the use of related works in system development can help improve the quality and effectiveness of software systems, and contribute to the ongoing evolution and advancement of the field.

**CHAPTER FOUR**

**System Analysis & Design**

# **Chapter Four System Analysis & Design**

System analysis and design shows the points of how the user will use the system. The application interaction diagrams needed to design the overall application attributes and methods and the interaction between different objects. These diagrams designed use case diagram, class diagrams, sequence diagrams and activity diagrams.

## **4.1 Functional view**

The functional view includes the following:

• Maintain basic data for the system.

• Transactions for the system.

### 4.1.1 Maintain basic data

In our system we have 2 basic data such as:

* Administrator (Name,Password)
* Network (Ip Router,Network Card)

### 4.1.2 Transactions for the system:

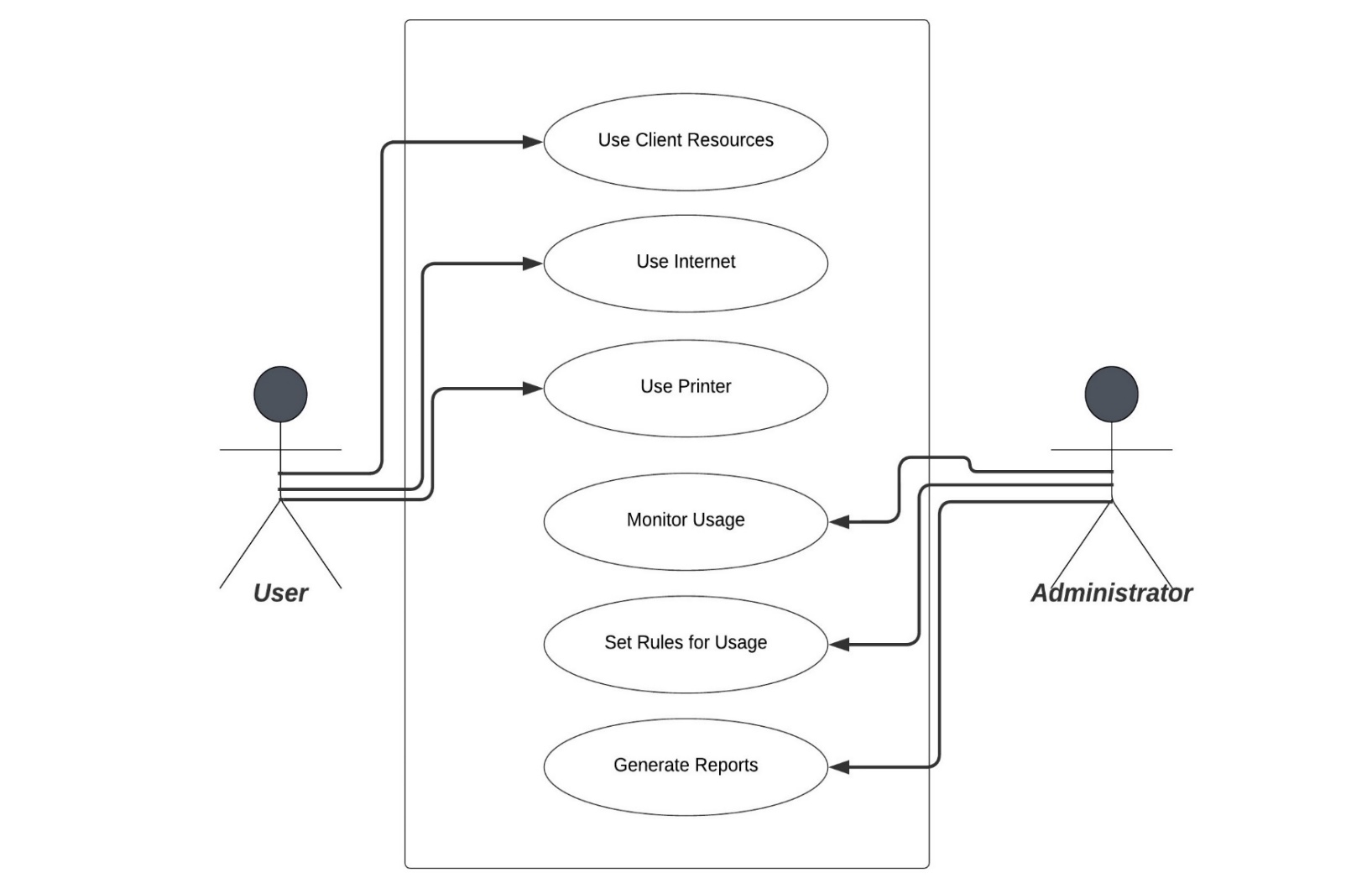
* **Login:** user login to the application after filling the email & password.
* **Scan Network:** Scan Network and ping on devices.
* **Internet Meters:** Check Network Speed download and upload.
* **Bandwidth:** Check Bandwidth of the Network.
* **Packet Sniffing:** technique whereby packet data flowing across the network is detected and observed.
* **Error Checker:** Check Errors in ping and send error to administrator.

## **Use case diagram**

is a graphical representation of the functional requirements of a software system or application. It shows the interactions between the system and its external actors, such as users, systems, or other entities. Use-case diagrams are typically used during the requirements gathering phase of software development to help identify and clarify system requirements and to communicate these requirements to stakeholders.

Use-case diagrams consist of a set of use cases, actors, and relationships between them. The use cases represent the system's functionalities or features, while the actors represent the external entities that interact with the system. The relationships between use cases and actors depict how the system interacts with its external environment.

### Use case diagram

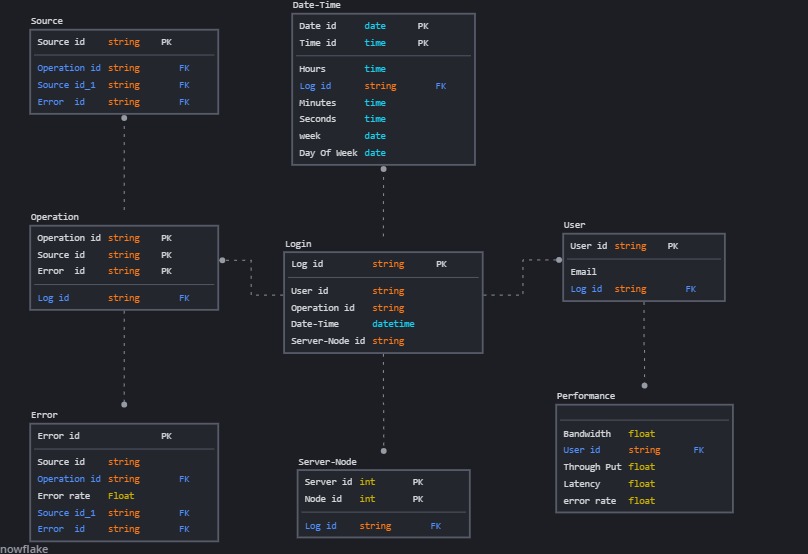


## **Database design using Class diagram**

Depicts the classes, interfaces, and relationships between them in a software system or application. A class diagram provides a high-level view of the system's structure, showing the classes and their attributes and methods, as well as the relationships and associations between them.

Classes are the building blocks of object-oriented programming, and a class diagram shows how they relate to each other in a software system or application. Each class in a class diagram represents a set of objects with similar attributes and behaviors. The attributes of a class are the data members, or variables, that describe the class, while the methods are the functions or operations that the class can perform.

### Class Diagram



### Sequence Diagram

shows the interactions between objects or components in a software system or application over time. A sequence diagram provides a visual representation of the flow of messages or events between the objects or components in the system, showing how they interact and communicate with each other.

In a sequence diagram, objects or components are represented as vertical lifelines, with messages or events represented as horizontal arrows between them. The sequence of messages or events is shown along the vertical axis of the diagram, with time increasing from top to bottom.

### 4.4.1 Sequence Diagram

## **Activity Diagram**

shows the flow of activities and actions in a software system or application. Activity diagrams provide a visual representation of the process flow, showing the sequence of activities, decisions, and actions that occur in the system over time.

In an activity diagram, activities are represented as rounded rectangles, with arrows connecting them to show the flow of the process. The arrows can be labeled with conditions or constraints that determine the flow of the process based on specific events or conditions.

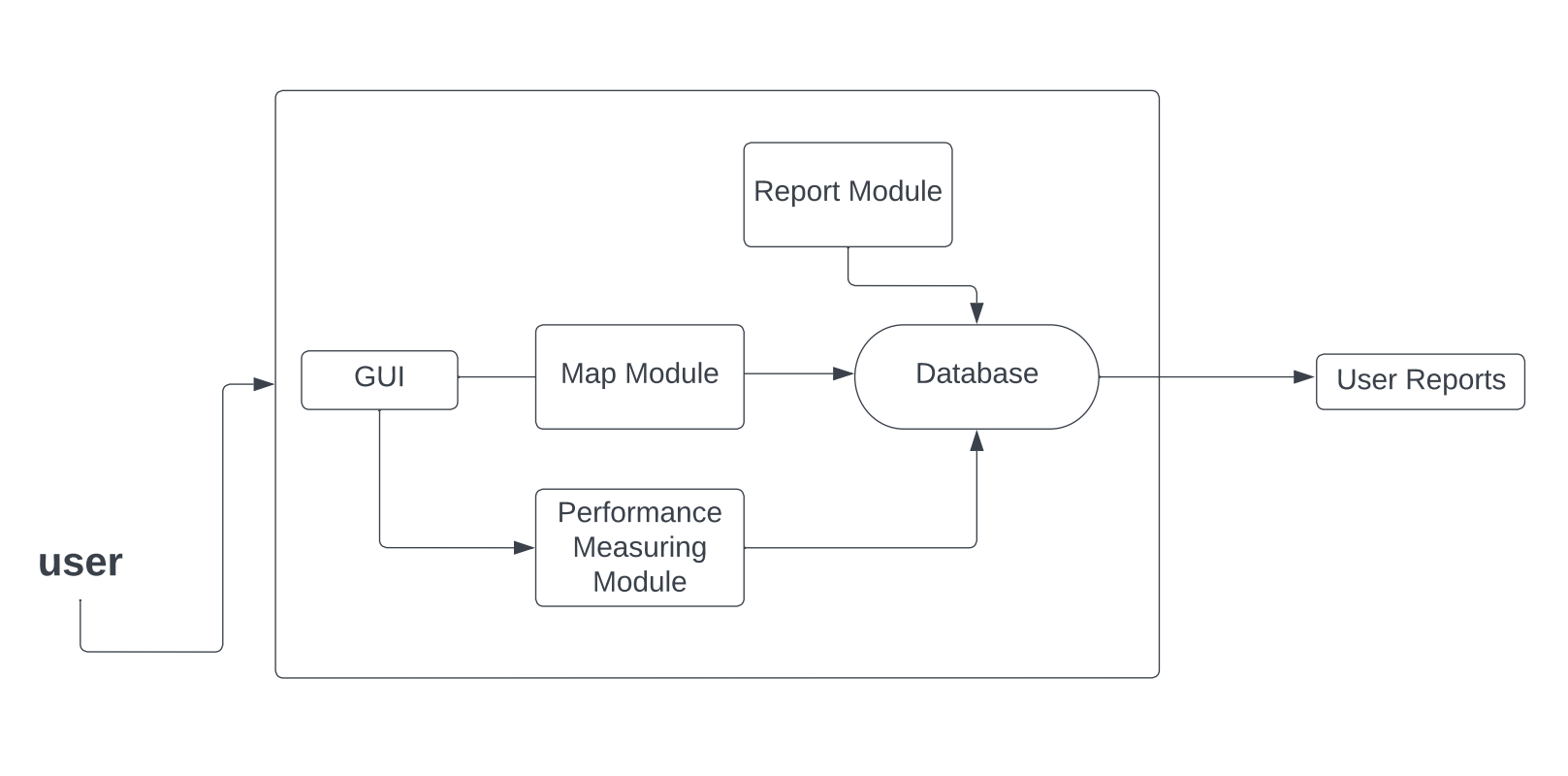
### 4.5.1 Activity Diagram

## **Block Diagram**

type of diagram that provides an overview of the components or subsystems of a system or process. It is used to represent a complex system or process in a simplified and visual way, showing the functional relationships between the different components or subsystems.

In a block diagram, each component or subsystem is represented as a rectangular block, with the input and output signals or data represented as arrows or lines connecting the blocks. The blocks may be connected in series or in parallel to represent the flow of data or signals between the components or subsystems.

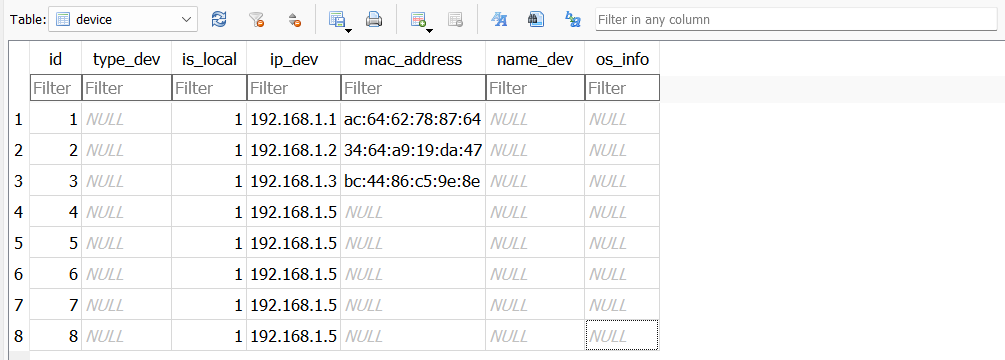
### Block Diagram



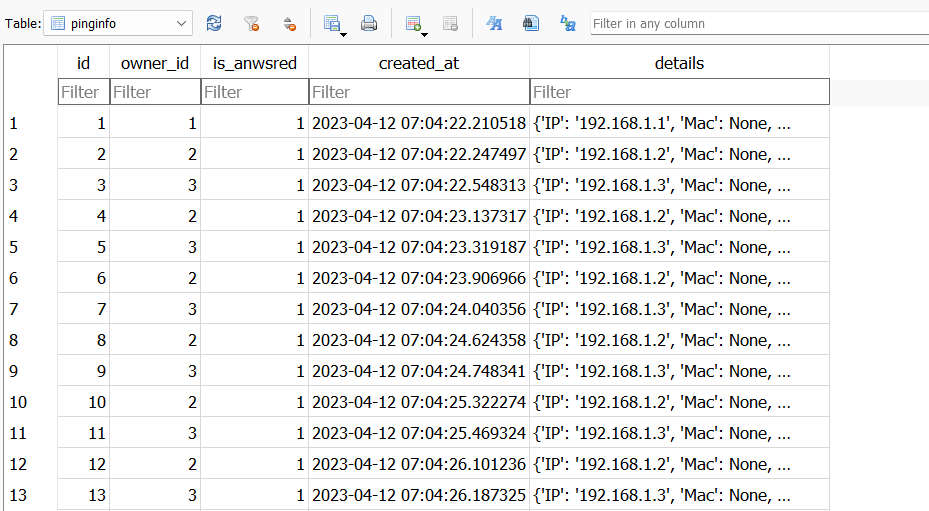
## **Database Tables**

Tables are used to store data in a structured and organized way, with each row representing a record and each column representing a field or attribute.

the columns represent the properties or attributes of the data stored in the table, and the rows represent individual instances or records of that data. Each column has a data type that defines the type of data that can be stored in that column, such as text, number, date, or boolean.



Devices Info – Table



Ping Info - Table

**CHAPTER FIVE**

**System Implementation**

# **Chapter Five System Implementation**

the process of building and deploying a software system or application, which involves translating the requirements and design specifications into executable code and deploying the system in a production environment.

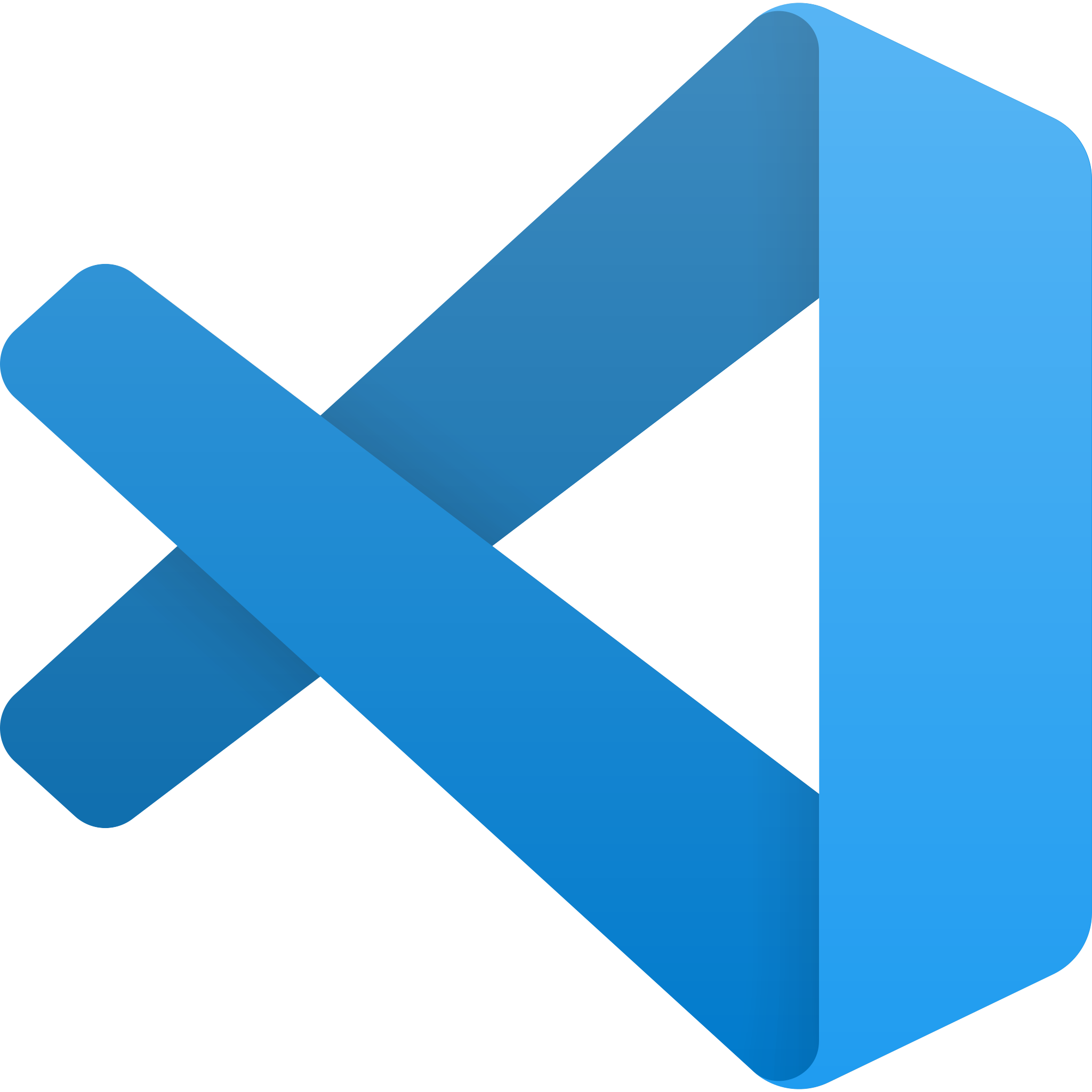
There is a detailed descriptive screenshot step by step of how we used these tools. There are the application’s screenshot and user interface are shown and descripted as well.

## **5.1 software & hardware used**

1. Visual Studio Code
2. QT Designer
3. Python V3

**The description of the software and hardware tools is written bellow:**

### 5.1.1 Visual Studio Code



also known as VS Code, is a source code editor developed by Microsoft. It is a free and open-source software that runs on Windows, Linux, and macOS platforms. VS Code is designed to be customizable and extensible, with support for a wide range of programming languages and frameworks.

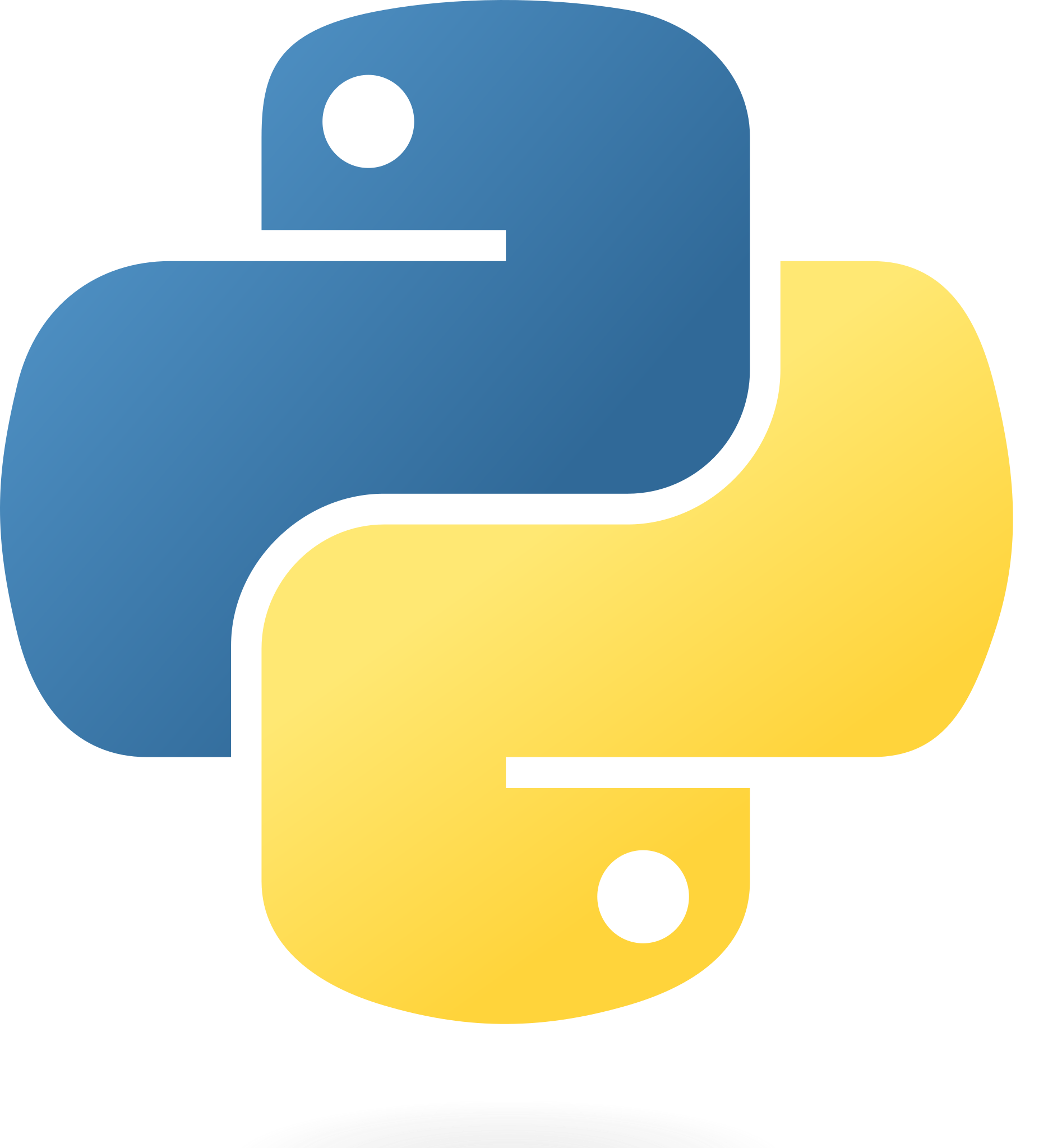
### 5.1.2 QT Designer



Graphical user interface (GUI) development tool that is used to create and design interfaces for QT-based applications. It is a part of the QT framework, which is a popular cross-platform application development framework.

QT Designer allows developers to create interfaces for their applications using drag-and-drop tools and a variety of widgets, such as buttons, labels, text boxes, and more. It also provides a library of pre-built widgets that can be customized and combined to create complex interfaces.

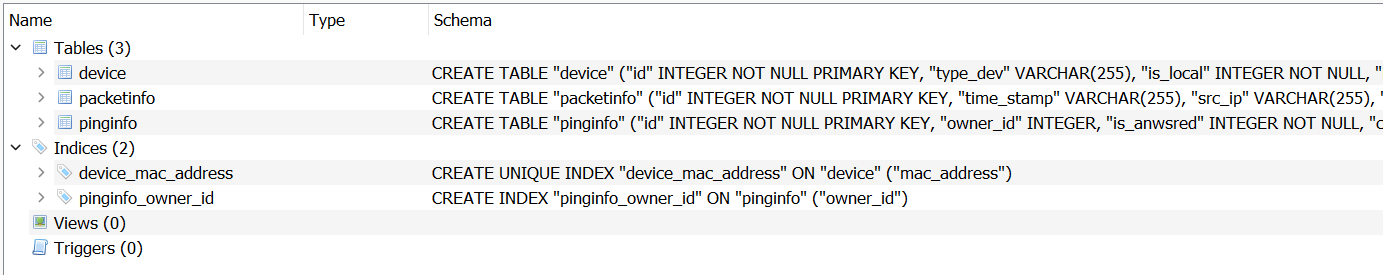
### 5.1.2 Python



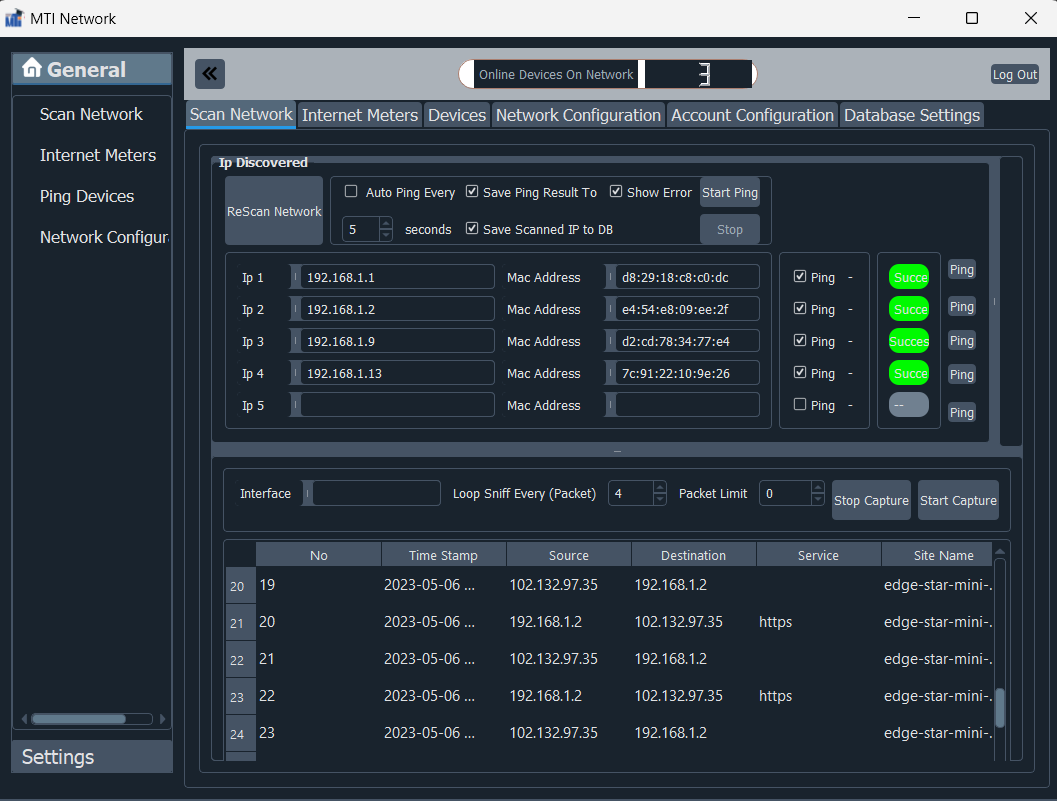
high-level, interpreted programming language that is widely used for a variety of applications, including web development, scientific computing, data analysis, artificial intelligence, and more. Python was first released in 1991, and it has since become one of the most popular programming languages in use today.

**5.2 Database implementation**

the process of creating and deploying a database system, which involves translating the database design into a physical database and setting up the necessary infrastructure to support the database.



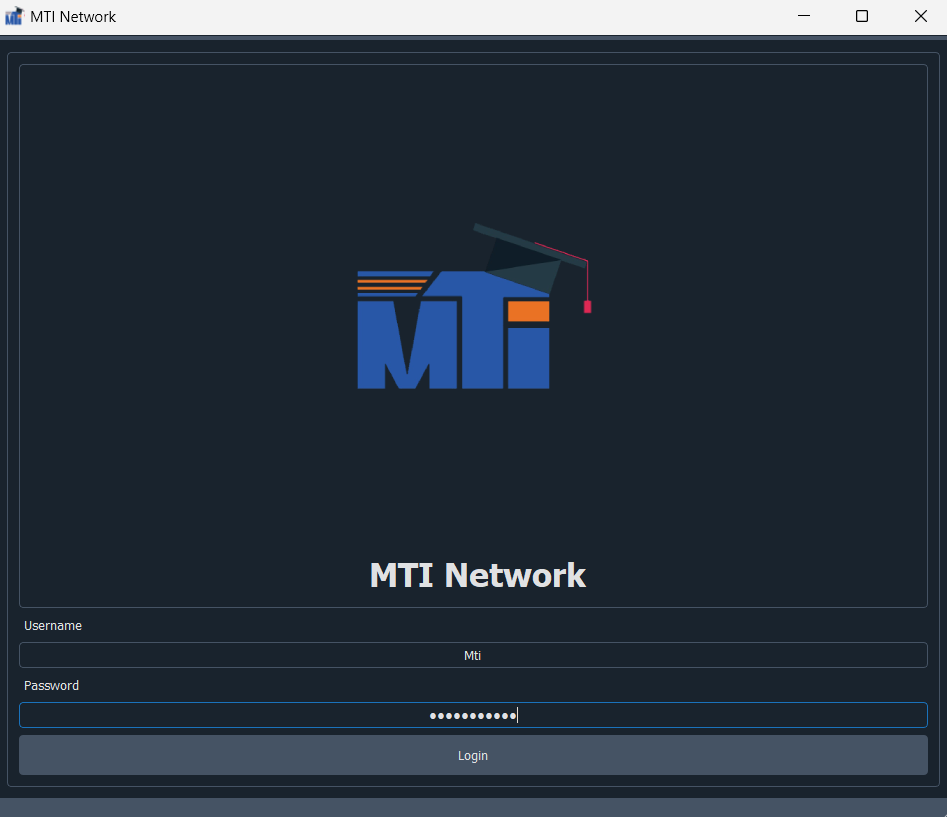
All database Tables



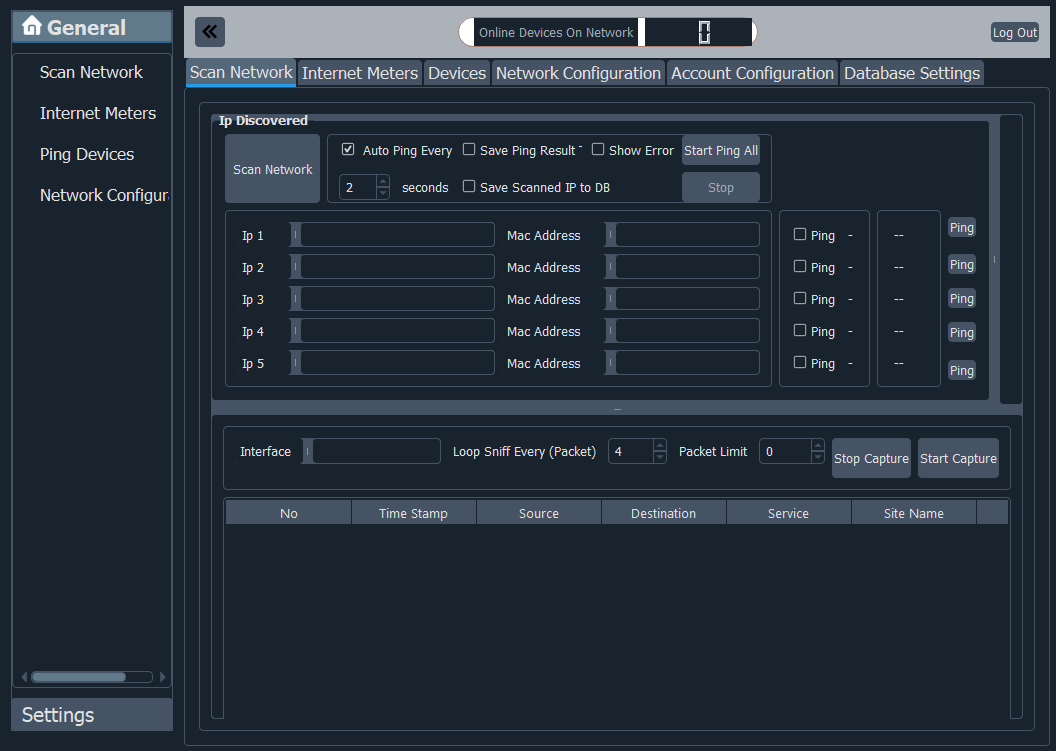
Packets Table

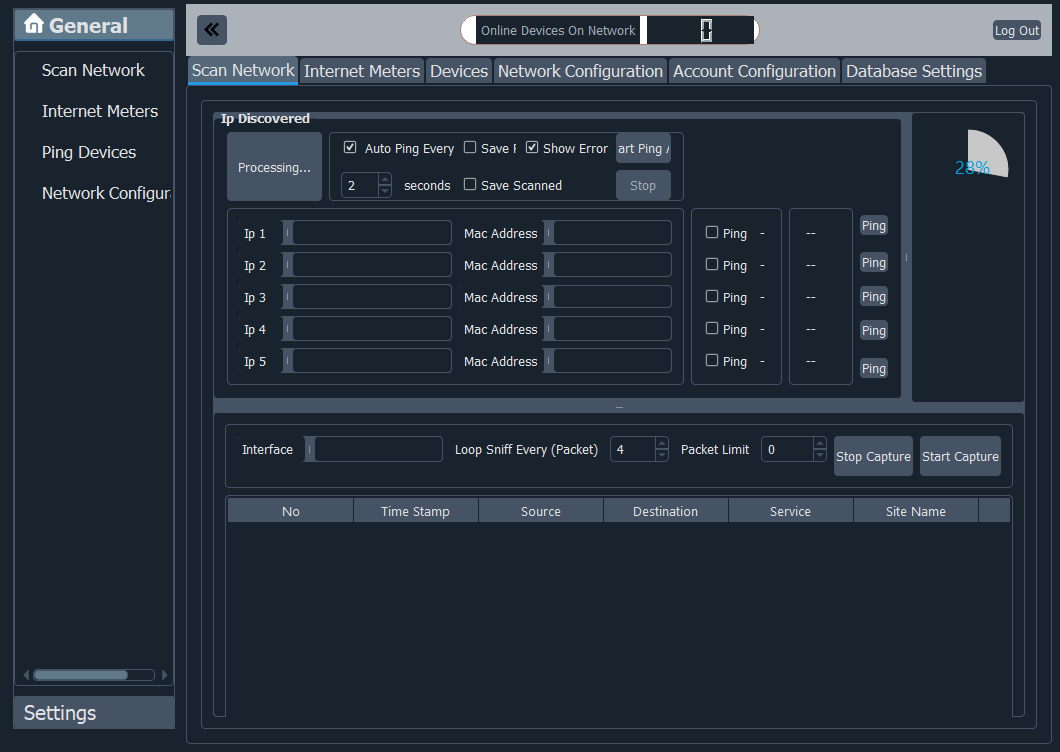
## **5.3 Graphical User Interface (GUI)**

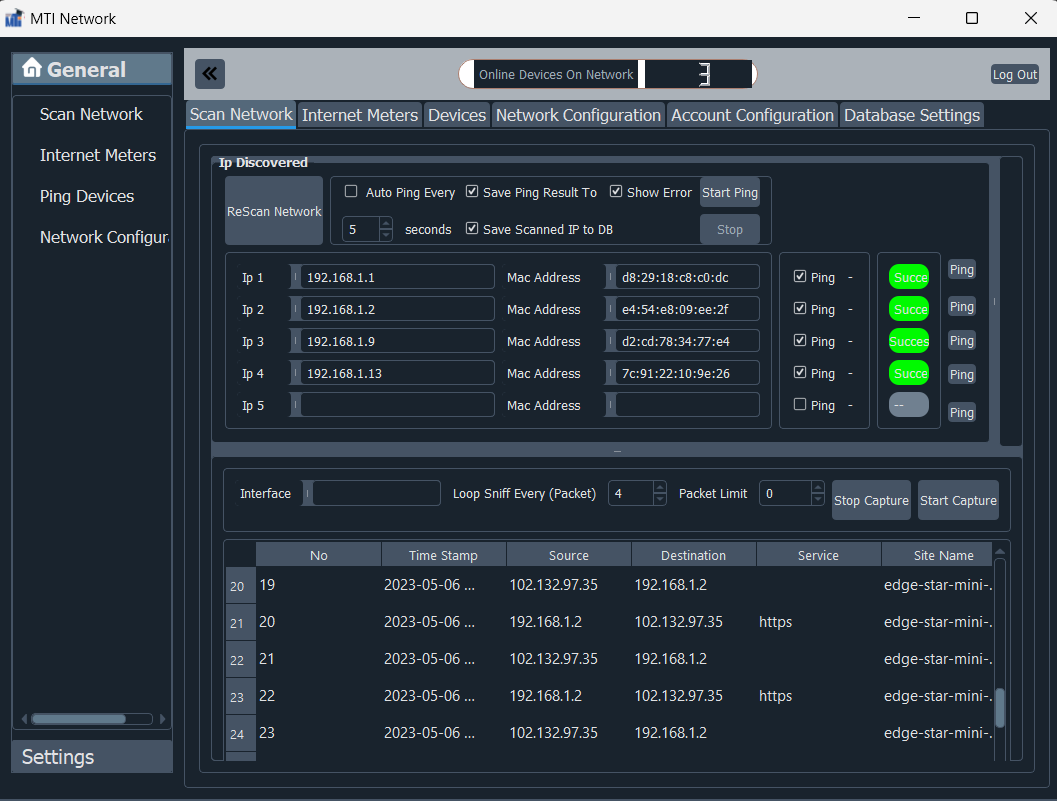
1. **Login:**



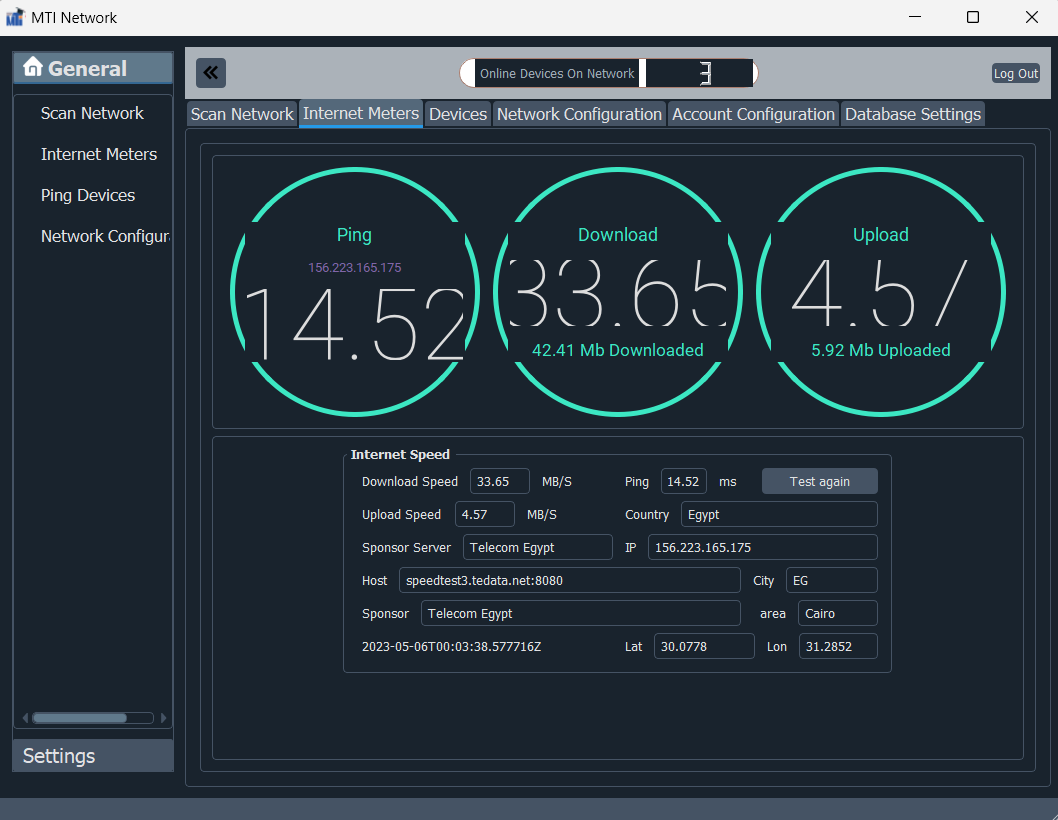
1. **Scan Network:**



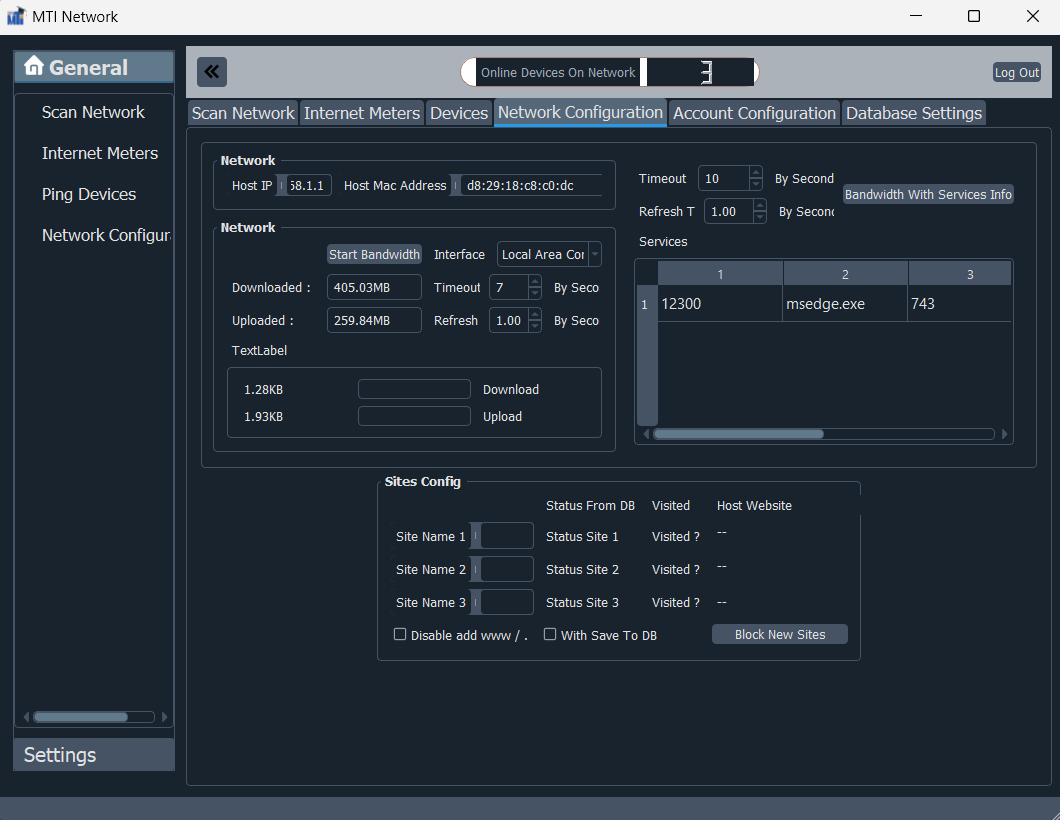




**3.Internet Meters:**

****

**4.Network Configuration:**

****

**CHAPTER SIX**

**System Testing**

# **Chapter Six System Testing**

## **6.1 Testing Importance**

System testing is the type of testing to check the behavior of a complete and fully

integrated software product based on the software requirements specification document.

The main focus of this testing is to evaluate Business / Functional / End-user requirements.

Purpose of user testing is to know the user’s feedback to ensure the usability of our system. By using a questionnaire to know the user’s opinion in our system by asking fifty of them to fill the given table after using our system by rating user interface, design, functionality, usefulness from scale 0 to 5 to know their opinion on our system.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Bad | Acceptable | Good | Very Good | Excellent |
| User Interface |  |  |  |  |  |
| Design |  |  |  |  |  |
| Functionality |  |  |  |  |  |
| Usefulness |  |  |  |  |  |

* User Interface: how easy the user navigates and use the interface?
* Design: the user’s opinion on the GUI layout
* Functionality: how good the system operates its functions?
* Usefulness: how good the users find our system useful?

## **6.2 System Testing Evaluation**

* Evaluation is made to determine how our system is qualified and how the user interacts with it.
* User testing provides what users think of the system. If the scores of user interface didn’t rate highly, then the program didn’t meet user’s expectations.
* If the functionality and the design were complicated and not easy to use; therefore, the user will not be satisfied using our system.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Bad | Acceptable | Good | Very Good | Excellent |
| User Interface |  |  |  | 🗹 |  |
| Design |  |  | 🗹 |  |  |
| Functionality |  |  |  | 🗹 |  |
| Usefulness |  |  |  |  | 🗹 |

* User Interface : System User Interface will be easily used by many users, and it can be improved in the next version.
* Design : System Design colors are good and easily to map.
* Functionality : System specifications are great and met requirements.
* Usefulness : The system is very useful for administrators or IT to specify any issue on the network.

## **6.3 Tests we performed**

### 6.3.1 Unit testing

by testing individual modules of an application in isolation (without any interaction with dependencies) to confirm that the code is doing things right.

6.3.2 Integration testing

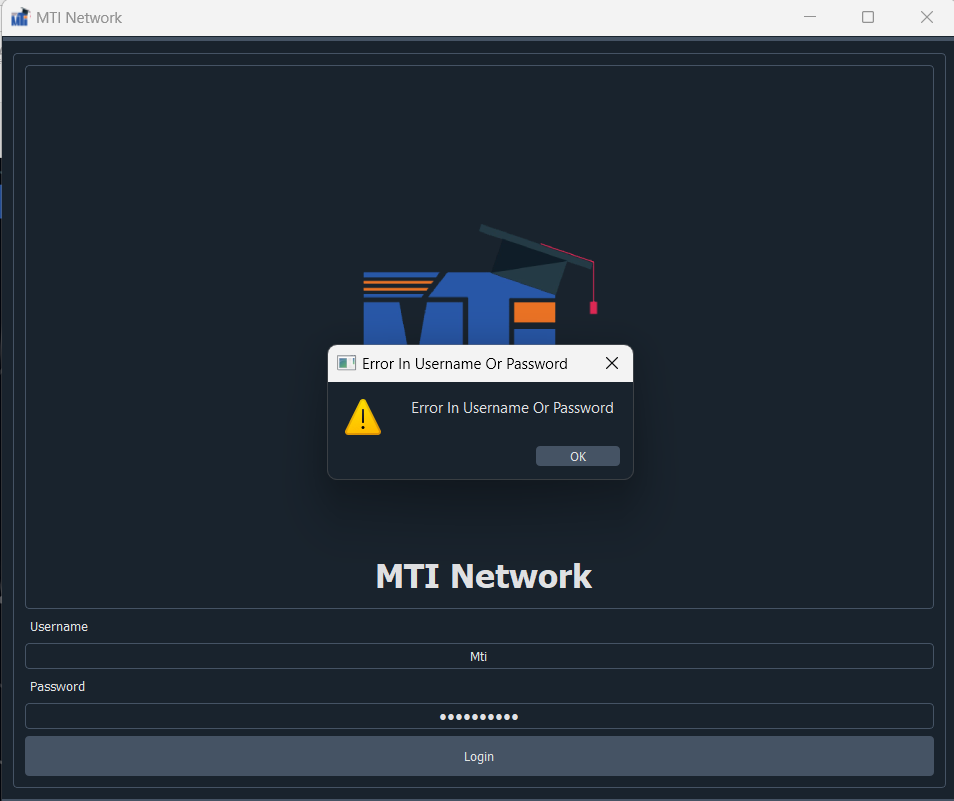
by checking if different modules are working fine when combined together as a group.

### 6.3.3 Test case scenario:

detailed description of a specific test case that outlines the objectives, steps, expected results, and test data needed to execute the test. It provides a clear and concise set of instructions for the tester to follow, ensuring that the test is executed consistently and accurately.

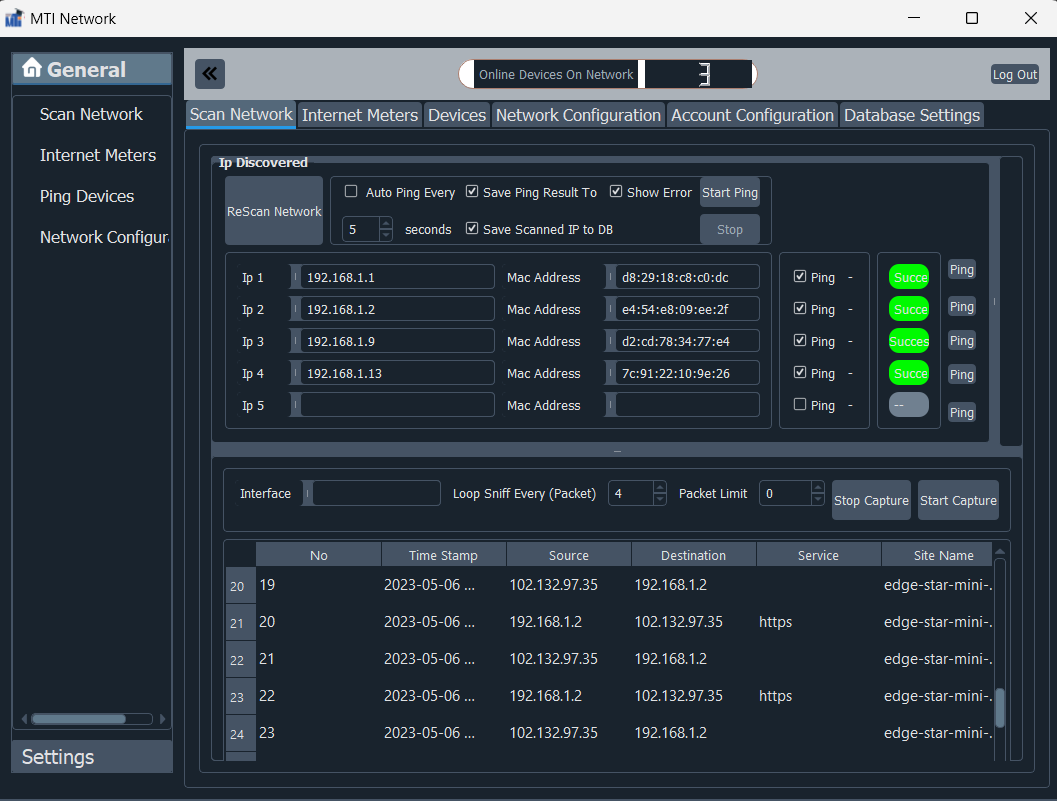
1. **User entering incorrect username or password**

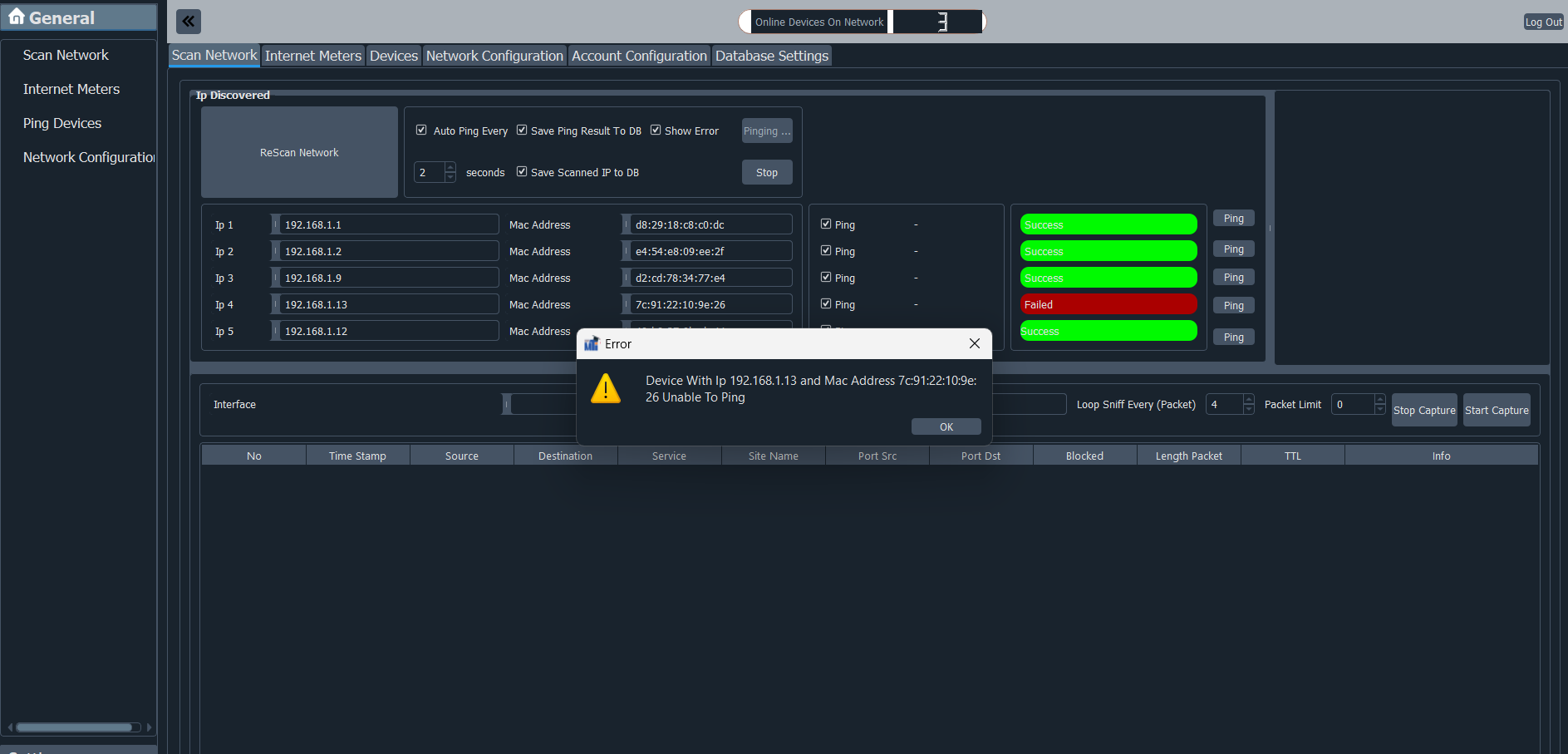
* Showed error message



1. **Scan Network & Ping Devices**

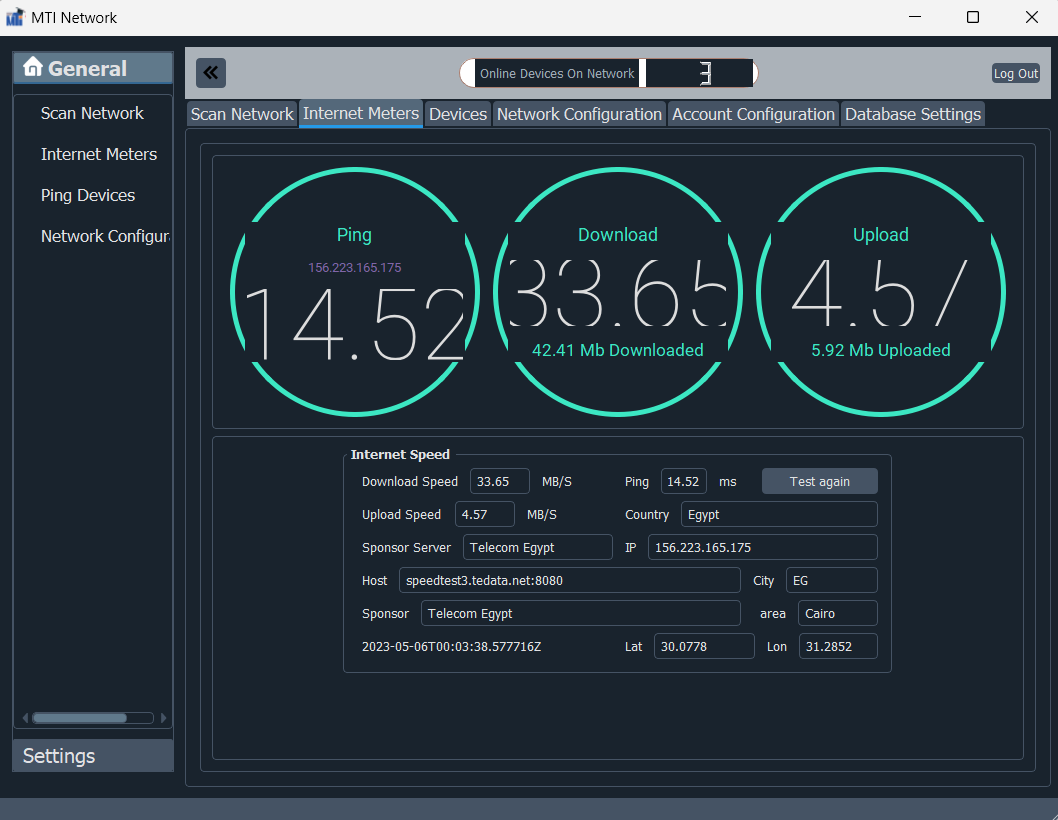
* **Showed connected networks and try to ping with them**

****

****

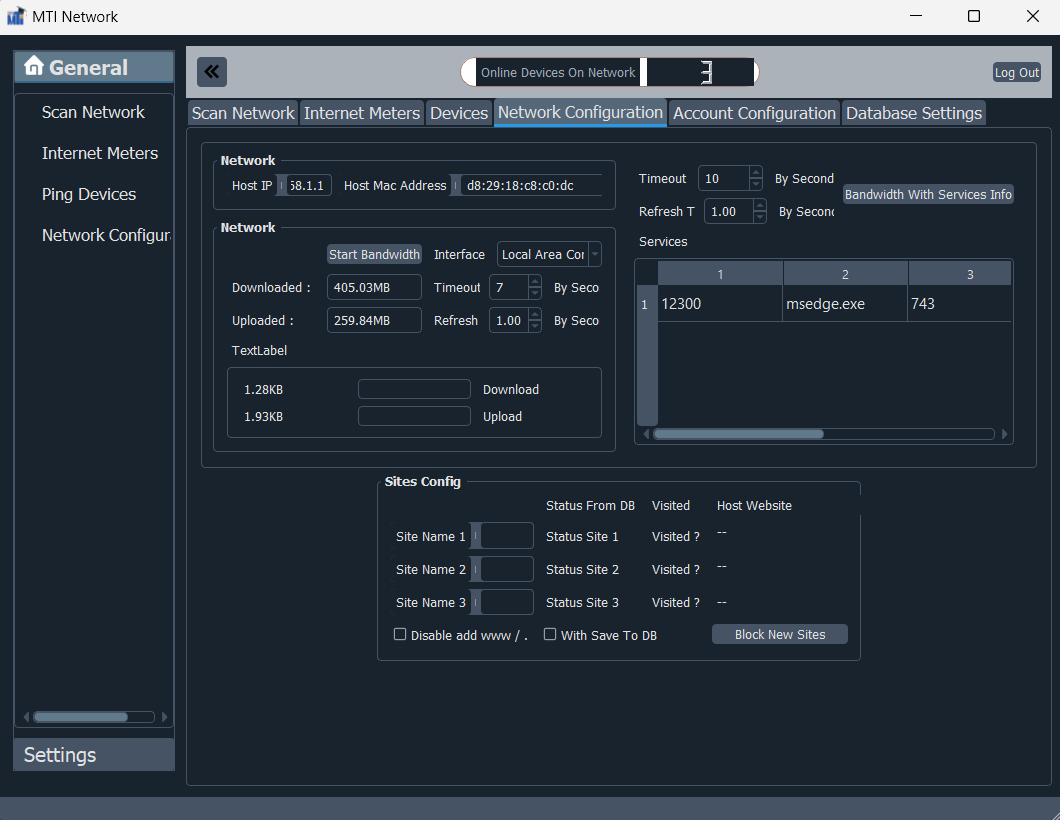
1. **Internet Meters**

* **Showed Internet Speed Download and upload**

****

**4-Bandwidth**

* **Showed bandwidth of network and services which consume it.**

****

**CHAPTER SEVEN**

**Conclusion**

**And Future Work**

# **Chapter Seven Conclusion And Future Work**

## **7.1 Conclusion**

a program used to monitor systems, analyze networks and check errors that occurred on them.it's easier for administrators to check failures on the network and check everything instantly.

**In this project we learned a lot of skills that improved our ability to work in teamwork and gained a lot of knowledge in the following areas:**

* Surveying Good Application met requirements and specifications
* Featuring the system requirements using UML diagrams and database designs
* Ability to use different implementation tools
* Discovered a lot of gathered information about the related systems
* The importance of system analysis UML diagrams
* How Teamwork can be powerful in developing large and effective applications
* How system testing makes sure that the system runs without errors or bugs

## **7.2 Future Work**

In addition to our work done it can be improved by adding some functions to improve our System such as the following:

* **More Improvements on UI**
* **More Support**
* **New Detection Methods**
* **More Specific Reports**

**CHAPTER EIGHT**

**References**

# **Chapter Eight References**

* [cisco](https://www.cisco.com)
* [solarwinds](https://www.solarwinds.com)
* [vmware](https://www.vmware.com)
* [avinetworks](https://avinetworks.com)
* [techtarget](https://www.techtarget.com/)
* [comparitech](http://www.comparitech.com)
* [infoabouttomorrow](http://www.infoabouttomorrow.com)

**CHAPTER NINE**

**Appendix**

# **Chapter Nine Appendix**

## **9.1 Program.py**

import sys

import os

import time

import datetime

import timeit

import subprocess

# import numpy as np

# from PyQt5.QtDesigner import \*

# from PyQt5.QtGui import \*

# from PyQt5.QtCore import \*

# from PyQt5 import QtWidgets as qtw

# from PyQt5.QtQml import QQmlApplicationEngine

# from PyQt5.uic import loadUiType

# Import As Pyside2

from PySide2 import QtUiTools

from PySide2.QtWidgets import \*

from PySide2.QtGui import \*

from PySide2.QtCore import \*

from PySide2 import QtWidgets as qtw

from PySide2.QtUiTools import loadUiType

# Import Network Modules

from scapy import all as sc

# import pyshark

# from front import resource\_rc

################ Import Custom ################

# from meters import AnalogGaugeWidget

from threads\_custom.scan\_network import scan\_network

## Import Buttons Config & Signal

from btns.circle\_btns import \*

from btns.btns import \*

## Import Threads Functions

from threads\_custom.works import Worker

## Test Speed Internet

from threads\_custom.test\_speed import \*

## Bandwidth Functions

from threads\_custom.bandwidth\_usage\_worker import \*

from threads\_custom.multi\_usage\_worker import \*

## Import Database Models

from db.my\_models import Device,PingInfo

## Import Database Functions

#from db.retrieve\_from\_db import get\_all\_dev

from db\_to\_qt import \*

from db.db\_config import db as my\_db

from matplot\_functions.set\_sample import \*

### Import Stylesheet Functions

#from qt\_material import apply\_stylesheet

import qdarkstyle

ui\_main,\_ = loadUiType('front/main\_window2.ui')

ui\_login,\_ot = loadUiType('front/second\_window.ui')

is\_program\_running = True

logo\_url = "front/logo\_small.png"

class Login(QMainWindow , ui\_login):

    def \_\_init\_\_(self):

        QMainWindow.\_\_init\_\_(self)

        self.setupUi(self)

        self.btn\_open = self.findChild(qtw.QPushButton,"pushButton")

        self.btn\_open.clicked.connect(self.log\_in\_system)

        self.setWindowIcon(QIcon(logo\_url))

        self.setWindowTitle("MTI Network")

        pixmap = QPixmap(f'{logo\_url}')

        self.label\_logo.setPixmap(pixmap)

        self.label\_logo.resize(pixmap.width(), pixmap.height())

    def log\_in\_system(self):

        valid\_auth = self.authenticate\_user()

        if valid\_auth == True:

            self.window2 = Main()

            self.close()

            self.window2.show()

        else:

            print("Wrong Password")

    def authenticate\_user(self):

        self.user = self.findChild(qtw.QLineEdit,"lineEdit\_user\_log").text()

        self.password = self.findChild(qtw.QLineEdit,"lineEdit\_password\_log").text()

        print(self.user)

        valid = False

        user\_name = "Mti"

        user\_pass = "network2023"

        if self.user == user\_name and self.password == user\_pass:

            valid = True

        else:

            s = "Error In Username Or Password"

            print(s)

            dlg = QMessageBox()

            dlg.setText(s)

            dlg.setIcon(QMessageBox.Warning)

            dlg.setStandardButtons(QMessageBox.Ok)

            dlg.setWindowTitle(s)

            button = dlg.exec\_()

        return valid

class Main(QMainWindow,ui\_main):

    def \_\_init\_\_(self):

        QMainWindow.\_\_init\_\_(self)

        self.setupUi(self)

        self.auto\_run()

        #self.threadsniffing()

    def auto\_run(self):

        print("auto Run Enabled")

        self.set\_vars()

        try:

            #print("Start To Connect Database ..")

            self.db.connect()

            #print("Connected To DB")

            #print("Closing Database ..")

            self.db.close()

            #print("Closed Connection To DB")

            print("Database Connected Success")

        except:

            print("Error To Connect Database")

        # \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

        #print("Maximum Threads : %d" % self.threadpool.maxThreadCount())

        # \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

        ## GUI Functions ##

        set\_btns(self)

        set\_circle\_design(self)

        reset\_circle\_val(self)

        setup\_btn(self)

        self.set\_table\_packet()

        # \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

        ## DataBase Functions ##

        self.set\_data\_base()

        #self.message\_error(s="Scan Network Please")

        # \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

        ## Custom Functions ##

        # networks\_cards = sc.get\_if\_list()

        # all\_network\_cards = sc.conf.ifaces

        # ip = sc.get\_if\_addr(sc.conf.iface)

        # print("IPPP", ip)

        # print("IFACES", all\_network\_cards)

        # for n in all\_network\_cards.items():

        #     print(n)

        #print(all\_network\_cards2)

        ## Thread For Scan Network For Ip & Mac Address .

        #self.threadRunner2()Realtek PCIe GBE Family Controller

        #self.threadRunner()

        ## Thread For Test Speed Internet and get Donwload , Upload Sppeed In MB/S , and other info .

        #self.threadRunner3()

        #self.threadRunner4()

        #self.set\_matplot()

    def open\_new\_win(self):

        print("Opened")

        self.win = Login()

        self.close()

        self.win.show()

    def set\_vars(self):

        ## For DB

        self.db = my\_db

        self.all\_devs = None

        ## For Sniff And Packet Show To Gui

        self.counter\_pkt = 0

        self.packet\_lista = []

        self.packet\_dict = {}

        ## For Block Sites

        self.ip\_blocked = {}

        ## Ping Vars

        self.error\_message = None

        ## Breaks

        self.LOOPPINGER = True

        self.LOOPTEST = True

        self.LOOPSCAN = True

        self.LOOPBNDWDTH = True

        self.LOOPSRVS= True

        self.SNF = True

        self.pkt\_num = 0

        self.threadpool = QThreadPool()

        #self.layout = qtw.QHBoxLayout()

        self.main\_widget = self.findChild(qtw.QTabWidget,"tabWidget")

        #print(dir(self.main\_widget))

        ## For Icon

        self.setWindowIcon(QIcon(logo\_url))

        self.setWindowTitle("MTI Network")

    def set\_data\_base(self):

        """

            Connect With Data Base And Config It,

            Get All quert and set it to variable

            Device = devs

            PingInfo = all\_pings

        """

        config\_data\_base(self)

        #all\_devs = collect\_database\_info()

    def config\_database\_finish(self):

        print("Data Collected From Database Finished")

        from db\_to\_qt import devss

        from db\_to\_qt import pingss

        #### Set To Gui Results Of Saved Records

        try:

            self.lineEdit\_database\_name.setText(str(db\_file\_name))

            self.lineEdit\_count\_ping.setText(str(len(pingss)))

            self.lineEdit\_count\_ping\_3.setText(str(len(pingss)))

            self.lineEdit\_count\_ip.setText(str(len(devss)))

            self.lineEdit\_count\_device.setText(str(len(devss)))

            print(len(devss))

            print(len(pingss))

        except:

            pass

    def set\_matplot(self):

        self.grafica = Canvas\_grafica()

        self.grafica1 = Canvas\_grafica2()

        self.grafica2 = Canvas\_grafica3()

        self.grafica3 = Canvas\_grafica4()

        self.grafica\_uno.addWidget(self.grafica)

        self.grafica\_dos.addWidget(self.grafica1)

        self.grafica\_tres.addWidget(self.grafica2)

        self.grafica\_cuatro.addWidget(self.grafica3)

    def set\_table\_packet(self):

        pos = self.tableWidget\_cap.verticalScrollBar().value()

        #print(len(self.packet\_lista))

        #self.tableWidget\_cap.setRowCount(1)

        table = QTableWidget()

        #qtw.QApplication.processEvents()

        self.tableWidget\_cap.verticalScrollBar().setValue(pos)

        self.tableWidget\_cap.horizontalHeader().setStretchLastSection(True)

    def insert\_to\_table(self):

        row\_packets = [{"no":"1","Time Stamp":"03:22"},{"no":"2","Time Stamp":"03:25"}]

        ####

        for num\_row, row\_pkt in enumerate(row\_packets):

            print(num\_row)

            print(row\_pkt)

            item\_no = qtw.QTableWidgetItem(row\_pkt['no'])

            item\_time = qtw.QTableWidgetItem(row\_pkt['Time Stamp'])

            ## setItem(row,column,str(item))

            #self.tableWidget\_cap.setItem(int,int,item)

            self.tableWidget\_cap.setItem(num\_row,0,item\_no)

            self.tableWidget\_cap.setItem(num\_row,1,item\_time)

    ########################## Start ##########################

    ###########     General methods for Network     ###########

    ###########################################################

    def ping\_ip(self,ip\_to\_ping):

        """

            Function To Ping Ip In \*args and return full response after ping with 3 packet as default send

        """

        command = ['ping','-n','1',ip\_to\_ping]

        return subprocess.getoutput(command)

    ## Functions For Buttons

    def which\_ip\_to\_ping(self,num\_ip):

        """

            Function Set For 5 Buttons In Main Window To Ping And Colored results

        """

        ip\_to\_ping = self.findChild(qtw.QLineEdit,f"lineEdit\_ip{str(num\_ip)}").text()

        mac\_addr = self.findChild(qtw.QLineEdit,f"lineEdit\_mac{str(num\_ip)}").text()

        ip\_ping\_response = {

            'IP':ip\_to\_ping,

            'Mac':mac\_addr,

            'Checked':False,

            'Response':False,

            'TTL':None,

            'ResTime':None,

        }

        try:

            ping\_response = self.ping\_ip(ip\_to\_ping)

            label\_to\_color = self.findChild(qtw.QLabel,f"label\_ping\_status\_ip{num\_ip}")

            if ip\_to\_ping == "" :

                label\_to\_color.setStyleSheet("background-color:slategray; border-radius:10px")

                #label\_to\_color.setStyleSheet(u"background-color: rgb(0, 0, 0);")

            else:

                if "Reply from" in ping\_response and \

                    not "Destination host unreachable." in ping\_response:

                    ### Set Label For Succes Ping

                    ## Color (In Stylesheet)

                    label\_to\_color.setStyleSheet("background-color:rgb(0,250,0); border-radius:10px")

                    ## Text (In setText)

                    label\_to\_color.setText("Success")

                    ## Find Results In Respons String In Terminal

                    ttl\_index = ping\_response.find("TTL=")

                    ttl = ping\_response[ttl\_index+4:ttl\_index+7].strip()

                    time\_index = ping\_response.find("time=")

                    time\_index\_end = ping\_response.find("ms TTL")

                    time\_ping = ping\_response[time\_index+5:time\_index\_end]

                    ## Set Results To variable ip\_ping\_response type(dict)

                    ip\_ping\_response['Checked'] = True

                    ip\_ping\_response['Response'] = True

                    ip\_ping\_response['TTL'] = ttl

                    if "time<1" in time\_ping:

                        time\_ping = "0"

                    ip\_ping\_response['ResTime'] = time\_ping

                else:

                    ip\_ping\_response['Checked'] = True

                    ip\_ping\_response['Response'] = False

                    label\_to\_color.setStyleSheet(u"background-color: rgb(170, 0, 0); border-radius:10px")

                    label\_to\_color.setText("Failed")

        except Exception as err:

            print("Failed To Ping" , err)

        finally:

            print(ip\_ping\_response)

            print("Ping Done")

            return ip\_ping\_response

    ########################## Start ##########################

    ### Methods For (Thread , Worker ) Instance and Objects ###

    ###########################################################

        # in this function we create our thread and run it

    def threadRunner(self):

        self.frame\_1\_circle\_progres.show()

        self.frame\_1\_circle\_progres.show()

        worker\_1 = Worker(self.first\_thread, num=1)# create our thread and give it a function as argument with its args

        worker\_1.signals.result.connect(self.first\_thread\_result) # connect result signal of our thread to thread\_result

        worker\_1.signals.finished.connect(self.first\_thread\_finished) # connect finish signal of our thread to thread\_complete

        self.threadpool.start(worker\_1) # start thread

    def first\_thread(self, num):

        print("first\_thread")

        # some long processing

        self.btn\_thread.setText("Processing...")

        #ip\_scanned = scan\_network("192.168.1.1/24")

        #print(ip\_scanned)

        ## Time Top Process

        time\_choosed = 20

        ## resault in %100

        stop\_point\_of\_100 = time\_choosed/100

        ## convert to msleep (sleep 1 second = mlsleep 1000 mlsecond)

        stop\_point\_of\_100 = stop\_point\_of\_100 \* 1000

        print(stop\_point\_of\_100)

        #QThread.sleep(2)

        num = 0

        for i in range(101):

            #get\_v = self.frame\_1\_circle\_progres.rpb\_getValue()

            if self.LOOPSCAN == True and i < 100:

                print(i)

                # if self.lineEdit\_ip\_host.text() != "":

                #     self.frame\_2\_circle\_progres.rpb\_setValue(100)

                #     break

                num += 1

                #val\_now = self.frame\_2\_cricle\_progres.rpb\_getValue()

                #print(val\_now)

                self.frame\_1\_circle\_progres.rpb\_setValue(i)

                QThread.msleep(stop\_point\_of\_100)

            if self.LOOPSCAN == False:

                print("Real Value Now",i)

                for n in range(i,101):

                    self.frame\_1\_circle\_progres.rpb\_setValue(n)

                    QThread.msleep(10)

                break

        return num

    def first\_thread\_result(self):

        #self.frame\_2\_circle\_progres.rpb\_setValue(100)

        print("First Thread Result")

        self.btn\_thread.setText("Finished")

        QThread.sleep(1)

    def first\_thread\_finished(self):

        print("First Thread Scan finished")

        self.btn\_thread.setText("ReScan Network")

        self.LOOPSCAN = True

        QThread.msleep(10)

        self.frame\_1\_circle\_progres.hide()

    def second\_thrd\_scan\_network(self):

        global ip\_scanned

        #self.btn\_thread.setText("Processing...")

        print("Start Scan")

        global ip\_router , ip\_master

        ip\_router = self.lineEdit\_ip\_router.text()

        ip\_master = self.lineEdit\_ip\_master.text()

        if ip\_master != "":

            pass

        else:

            ip\_master = None

        if ip\_router != "":

            ip\_scanned = scan\_network(f"{str(ip\_router)}/24")

        else:

            ip\_router = None

            ip\_scanned = scan\_network("192.168.1.1/24")

        #print("Finish Scan")

        #print(ip\_scanned)

        return ip\_scanned

    def third\_thrd\_speed\_test(self):

        self.frame\_3\_circle\_progres.show()

        self.pushButton\_test\_speed.setText("Processing...")

        print("Start Test Speed")

        global speed\_test\_dict

        st = speedtest.Speedtest()

        st.get\_best\_server()

        download\_speed = return\_bytes\_by\_mb(st.download())

        self.LOOPTESTDN = False

        uploaad\_speed = return\_bytes\_by\_mb(st.upload())

        self.LOOPTESTUP = False

        speed\_test\_dict = st.results.dict()

        downloaded = return\_bytes\_by\_mb(speed\_test\_dict['bytes\_received'])

        print(speed\_test\_dict)

        print(downloaded)

        #print("Your Download speed is", download\_speed, "MB")

        print("Finished Test Speed Internet")

        #print("End Result First Thread")

        self.LOOPTEST = False

        return speed\_test\_dict

    def fourth\_thrd\_speed\_test(self):

        print("Start Fourth Thread Speed Test Progress")

        time\_choosed = 25

        ## resault in %100

        stop\_point\_of\_100 = time\_choosed/100

        ## convert to msleep (sleep 1 second = mlsleep 1000 mlsecond)

        stop\_point\_of\_100 = stop\_point\_of\_100 \* 1000

        print(stop\_point\_of\_100)

        #QThread.sleep(2)

        for i in range(101):

            if self.LOOPTEST == True:

                print(i)

                #self.frame\_1\_circle\_progres.rpb\_setValue(i)

                self.frame\_3\_circle\_progres.rpb\_setValue(i)

                QThread.msleep(stop\_point\_of\_100)

            else:

                self.frame\_3\_circle\_progres.rpb\_setValue(i)

                QThread.msleep(10)

    def second\_thread\_scan\_result(self):

        """

            Scan Network Results

            Loop on IP Scanned for Collect Info Stored In Dictionary (global var) > Updated in method : self.thread\_scan\_network

        """

        self.LOOPSCAN = False

        self.section\_network.setEnabled(True)

        self.section\_ip\_discovered.setEnabled(True)

        for n,info in ip\_scanned.items():

            print(info)

            if info['IP'] == ip\_master:

                info['master'] = True

            else:

                info['master'] = False

            if info['IP'] == ip\_router:

                info['host'] = True

            else:

                info['host'] = False

        print(len(ip\_scanned))

        for row\_num , row\_info in ip\_scanned.items():

            print(row\_num)

            print(row\_info)

            ip = row\_info['IP']

            mac = row\_info['Mac']

            if row\_info['host'] == True:

                label\_ip = self.findChild(qtw.QLineEdit,"lineEdit\_ip\_host")

                label\_mac = self.findChild(qtw.QLineEdit,"lineEdit\_mac\_host")

                label\_ip.setText(str(ip))

                label\_mac.setText(str(mac))

            check\_ip = self.findChild(qtw.QCheckBox,f'checkBox\_ping\_{str(row\_num)}')

            label\_ip = self.findChild(qtw.QLineEdit,f"lineEdit\_ip{str(row\_num)}")

            label\_mac = self.findChild(qtw.QLineEdit,f"lineEdit\_mac{str(row\_num)}")

            #print(dir(self.label\_ip))

            label\_ip.setText(str(ip))

            label\_mac.setText(str(mac))

            check\_ip.setChecked(True)

    def third\_thrd\_speed\_test\_result(self):

        print("Third Thread Scan Result")

        self.pushButton\_test\_speed.setText("Test again")

    def fourth\_thrd\_speed\_test\_result(self):

        print("Fourth Thread Speed Test Progress Result")

        self.LOOPTEST = True

        self.frame\_3\_circle\_progres.hide()

    def second\_thread\_scan\_finished(self):

        print("Second Thread Scan finished")

    def third\_thrd\_speed\_test\_finished(self):

        print("Third Thread Speed Test Finished")

        down\_speed\_mb = return\_bytes\_by\_mb(speed\_test\_dict['download'])

        up\_speed\_mb = return\_bytes\_by\_mb(speed\_test\_dict['upload'])

        time\_stamp = speed\_test\_dict['timestamp']

        all\_uploaded = return\_bytes\_by\_mb(speed\_test\_dict['bytes\_sent'])

        all\_downloaded = return\_bytes\_by\_mb(speed\_test\_dict['bytes\_received'])

        try:

            time\_stamp = round(time\_stamp,2)

            print(time\_stamp)

        except:

            pass

        # Convert To Limit 2 After ,

        dwn\_mb = round(down\_speed\_mb,2)

        up\_mb = round(up\_speed\_mb,2)

        all\_up = str(round(all\_uploaded,2)) + " Mb Uploaded"

        all\_down = str(round(all\_downloaded,2)) + " Mb Downloaded"

        ping = round(speed\_test\_dict['ping'],2)

        print(time\_stamp)

        print(all\_up)

        print(all\_down)

        # Set To Gui

        self.lineEdit\_download.setText(str(dwn\_mb))

        self.label\_download\_meter.setText(str(dwn\_mb))

        self.lineEdit\_upload.setText(str(up\_mb))

        self.label\_upload\_meter.setText(str(up\_mb))

        self.lineEdit\_ping.setText(str(ping))

        self.label\_meter\_ping.setText(str(ping))

        self.lineEdit\_country.setText(str(speed\_test\_dict['server']['country']))

        self.lineEdit\_host.setText(str(speed\_test\_dict['server']['host']))

        self.lineEdit\_sponsor\_server.setText(str(speed\_test\_dict['server']['sponsor']))

        self.lineEdit\_area.setText(str(speed\_test\_dict['server']['name']))

        self.lineEdit\_city.setText(str(speed\_test\_dict['client']['country']))

        self.lineEdit\_sponsor.setText(str(speed\_test\_dict['client']['isp']))

        self.lineEdit\_lon.setText(str(speed\_test\_dict['client']['lon']))

        self.lineEdit\_lat.setText(str(speed\_test\_dict['client']['lat']))

        self.lineEdit\_global\_ip.setText(str(speed\_test\_dict['client']['ip']))

        self.label\_all\_up.setText(str(all\_up))

        self.label\_all\_down.setText(str(all\_down))

        self.label\_test\_time.setText(str(time\_stamp))

        self.label\_meter\_ip.setText(str(speed\_test\_dict['client']['ip']))

    def fourth\_thrd\_speed\_test\_finished(self):

        print("Fourth Thread Speed Test Finished")

    def threadRunner4(self):

        worker\_1 = Worker(self.fourth\_thrd\_speed\_test)# create our thread and give it a function as argument with its args

        worker\_1.signals.result.connect(self.fourth\_thrd\_speed\_test\_result) # connect result signal of our thread to thread\_result

        worker\_1.signals.finished.connect(self.fourth\_thrd\_speed\_test\_finished) # connect finish signal of our thread to thread\_complete

        self.threadpool.start(worker\_1) # start thread

    def threadRunner3(self):

        worker\_1 = Worker(self.third\_thrd\_speed\_test)# create our thread and give it a function as argument with its args

        worker\_1.signals.result.connect(self.third\_thrd\_speed\_test\_result) # connect result signal of our thread to thread\_result

        worker\_1.signals.finished.connect(self.third\_thrd\_speed\_test\_finished) # connect finish signal of our thread to thread\_complete

        self.threadpool.start(worker\_1) # start thread

    def threadRunner2(self):

        worker\_2 = Worker(self.second\_thrd\_scan\_network) # create our thread and give it a function as argument with its args

        worker\_2.signals.result.connect(self.second\_thread\_scan\_result) # connect result signal of our thread to thread\_result

        worker\_2.signals.finished.connect(self.second\_thread\_scan\_finished) # connect finish signal of our thread to thread\_complete

        self.threadpool.start(worker\_2)

    def threadsniffing(self):

        worker = Worker(self.sniffer\_start)

        worker.signals.result.connect(self.sniffer\_results)

        worker.signals.finished.connect(self.sniffer\_finished)

        self.threadpool.start(worker)

    ###### Network Functions Custom for re view results catched

    def get\_serv(self,src\_port,dst\_port):

        try:

            service = socket.getservbyport(src\_port)

        except:

            service = socket.getservbyport(dst\_port)

            return service

    def get\_host\_name(self,ip):

        if "192.168" in ip:

            name = "Local Device"

        else:

            try:

                name = socket.getfqdn(ip)

            except:

                name = None

        return name

    def analyzer\_sniff(self,pkt):

        ## packet dictionary

        # num

        # time

        # ip-src

        # ip-dst

        # mac-src

        # mac-dst

        # port-src

        # port-dst

        is\_blocked = False

        try:

            ## Full Date

            ## strftime("%H:%M:%S")

            dt\_object = datetime.fromtimestamp(pkt.time)

            src = pkt.src

            dst = pkt.dst

            ttl\_pkt = pkt.ttl

            len\_pkt = pkt.len

            try:

                ######

                src\_ip = pkt[IP].src

                dst\_ip = pkt[IP].dst

                ########################

                mac\_src = pkt.src

                mac\_dst = pkt.dst

            except:

                src\_ip = None

                dst\_ip = None

                pass

            ## Get Host NAme

            host\_name = None

            host\_name1 = None

            host\_name2 = None

            try:

                if src\_ip != None and dst\_ip != None:

                    host\_name1 = self.get\_host\_name(ip=src\_ip)

                    host\_name2 = self.get\_host\_name(ip=dst\_ip)

            except Exception as err:

                print(err)

                pass

            try:

                if host\_name1 != None:

                    if host\_name1 != "Local Device":

                        host\_name = host\_name1

                        if host\_name1 == self.ip\_blocked[1]['host'] or self.ip\_blocked[1]['name'] in host\_name1 :

                            is\_blocked = True

                            print("Blocked")

                if host\_name2 != None:

                    if host\_name2 != "Local Device":

                        host\_name = host\_name2

                        if host\_name2 == self.ip\_blocked[1]['host'] or self.ip\_blocked[1]['name'] in host\_name2 :

                            is\_blocked = True

                            print("Blocked")

            except:

                pass

            self.pkt\_num += 1

            pckt\_dict = {

                'num':self.pkt\_num,

                'time':dt\_object,

                'ip-src':src,

                'ip-dst':dst,

                "ttl-pkt":ttl\_pkt,

                "len-pkt":len\_pkt,

                'port-src':None,

                'port-dst':None,

                'host-name':host\_name,

                'service':None,

                'blocked': is\_blocked,

            }

            #print(pckt\_dict)

            if pkt.haslayer(ICMP):

                print("----------------------------------------")

                #print("ICMP PACKET..." , "src-ip",src\_ip,"dst-ip",dst\_ip)

                #print("SRC-MAC : " + mac\_src)

                #print("DST-MAC : " + mac\_dst)

                if pkt.haslayer(Raw):

                    data = pkt[Raw].load

                ####

                pckt\_dict = {

                    'num':self.pkt\_num,

                    'layer':"ICMP",

                    'time':dt\_object,

                    'ip-src':pkt['IP'].src,

                    'ip-dst':pkt['IP'].dst,

                    "ttl-pkt":ttl\_pkt,

                    "len-pkt":len\_pkt,

                    'port-src':None,

                    'port-dst':None,

                    'host-name':host\_name,

                    'service':None,

                    'blocked': is\_blocked,

                    }

                #self.packet\_lista.append(pckt\_dict['num'])

                #self.set\_pkt\_on\_table(pckt\_dict)

            else:

                try:

                    src\_port = pkt.sport

                    dst\_port = pkt.dport

                    service  = self.get\_serv(src\_port,dst\_port)

                    pckt\_dict = {

                        'num':self.pkt\_num,

                        'time':dt\_object,

                        'ip-src':pkt['IP'].src,

                        'ip-dst':pkt['IP'].dst,

                        "ttl-pkt":ttl\_pkt,

                        "len-pkt":len\_pkt,

                        'port-src':str(src\_port),

                        'port-dst':str(dst\_port),

                        'host-name':host\_name,

                        'service':service,

                        'blocked':is\_blocked,

                        }

                    #print("Service",service)

                    if pkt.haslayer(TCP):

                        print("----------------------------------------")

                        print("TCP PACKET..." , "src-ip",src\_ip,'port-src',src\_port,"dst-ip",dst\_ip,'port-dst',dst\_port,"service",service)

                    if pkt.haslayer(UDP):

                        print("----------------------------------------")

                        print("UDP PACKET..." , "src-ip",src\_ip,"dst-ip",dst\_ip,"service",service)

                except:

                    pass

            if self.checkBox\_show\_packet.isChecked():

                print("Printing Packet To Table")

                self.set\_pkt\_on\_table(pckt\_dict)

                #self.packet\_dict[pkt.num]

        except Exception as err:

            print(err)

            pass

    def set\_pkt\_on\_table(self,pkt\_dict):

        try:

            print("Packet Insert To Table Widget")

            row = pkt\_dict['num']

            print(row)

            self.tableWidget\_cap.setRowCount(row + 1)

            item\_pkt\_no = qtw.QTableWidgetItem(str(pkt\_dict['num']))

            item\_pkt\_time = qtw.QTableWidgetItem(str(pkt\_dict['time']))

            item\_pkt\_ip\_src = qtw.QTableWidgetItem(pkt\_dict['ip-src'])

            item\_pkt\_ip\_dst = qtw.QTableWidgetItem(pkt\_dict['ip-dst'])

            item\_pkt\_len = qtw.QTableWidgetItem(pkt\_dict['len-pkt'])

            item\_pkt\_ttl = qtw.QTableWidgetItem(pkt\_dict['ttl-pkt'])

            self.tableWidget\_cap.setItem(row,0,item\_pkt\_no)

            self.tableWidget\_cap.setItem(row,1,item\_pkt\_time)

            self.tableWidget\_cap.setItem(row,2,item\_pkt\_ip\_src)

            self.tableWidget\_cap.setItem(row,3,item\_pkt\_ip\_dst)

            self.tableWidget\_cap.setItem(row,9,item\_pkt\_len)

            self.tableWidget\_cap.setItem(row,10,item\_pkt\_ttl)

            if pkt\_dict['service'] != None:

                item\_pkt\_service = qtw.QTableWidgetItem(pkt\_dict['service'])

                self.tableWidget\_cap.setItem(row,4,item\_pkt\_service)

            if pkt\_dict['host-name'] != None:

                item\_pkt\_host\_name = qtw.QTableWidgetItem(pkt\_dict['host-name'])

                self.tableWidget\_cap.setItem(row,5,item\_pkt\_host\_name)

            if pkt\_dict['port-src'] != None:

                item\_pkt\_port\_src = qtw.QTableWidgetItem(pkt\_dict['port-src'])

                self.tableWidget\_cap.setItem(row,6,item\_pkt\_port\_src)

            if pkt\_dict['port-dst'] != None:

                item\_pkt\_port\_dst = qtw.QTableWidgetItem(pkt\_dict['port-dst'])

                self.tableWidget\_cap.setItem(row,7,item\_pkt\_port\_dst)

            if pkt\_dict['blocked'] != False:

                item\_pkt\_blocked = qtw.QTableWidgetItem("Blocked Site")

                self.tableWidget\_cap.setItem(row,8,item\_pkt\_blocked)

        except Exception as err:

            print(err)

    def sniffer\_start(self):

        print("Start Sniffer")

        count\_pkt = self.spinBox\_pkt\_count\_by.text()

        count\_pkt = int(count\_pkt)

        limit\_pkt = self.spinBox\_pkt\_limit.text()

        print(limit\_pkt)

        self.counter\_pkt = 0

        limit\_pkt = int(limit\_pkt)

        if count\_pkt == 0:

            counter = 0

        else:

            counter = count\_pkt

        while self.SNF == True:

            sc.sniff(iface="Ethernet",prn=self.analyzer\_sniff,count=count\_pkt)

            counter += count\_pkt

            self.counter\_pkt = counter

            print("Packet Lista >>",counter)

            if limit\_pkt == 0:

                pass

            else:

                if counter >= limit\_pkt:

                    self.SNF = False

    def sniffer\_results(self):

        print("Results cap")

    def sniffer\_finished(self):

        save\_to\_db = self.checkBox\_save\_packet.isChecked()

        if save\_to\_db:

            print("Saving To DB")

            return

        else:

            print("Finished Sniffer")

    def stop\_snifer(self):

        self.SNF = False

        print("Stopped Sniff")

        print(len(self.packet\_lista) ," Packet Captured" )

        return self.SNF

    def continue\_snifer(self):

        self.SNF = True

        print(self.SNF)

        self.threadsniffing()

        return self.SNF

    #####################################

    ####### Sites Blocked Threads #######

    def block\_site\_worker(self):

        worker = Worker(self.start\_block\_site\_sample)

        self.threadpool.start(worker)

    def start\_block\_site\_sample(self):

        """

            BlockSite Functions Start Thread

        """

        strt\_bndwds = "Start Blocked Site Thread"

        print(strt\_bndwds)

        with\_out\_www = self.checkBox\_disable\_www.isChecked()

        site\_1 = self.lineEdit\_site\_name1.text()

        if site\_1 != "":

            if not with\_out\_www:

                site\_name1 = "www." + str(site\_1) + ".com"

            else:

                site\_name1 = str(site\_1)

            ip\_1 = socket.gethostbyname(site\_name1)

            host\_1 = socket.getfqdn(ip\_1)

        else:

            site\_1 = None

        site\_2 = self.lineEdit\_site\_name2.text()

        if site\_2 != "":

            if not with\_out\_www:

                site\_name2 = "www." + str(site\_2) + ".com"

            else:

                site\_name2 = str(site\_2)

            ip\_2 = socket.gethostbyname(site\_name2)

            host\_2 = socket.getfqdn(ip\_2)

        else:

            site\_2 = None

        #site\_3 = self.lineEdit\_site\_name3.text()

        #host\_2 = socket.gethostbyaddr(str(ip\_2))

        if site\_1 != None:

            self.ip\_blocked = {

                1 : {'ip' : ip\_1, 'name':site\_1 , 'host':host\_1} ,

            }

        if site\_2 != None:

            self.ip\_blocked = {

                2 : {'ip' : ip\_1, 'name':site\_1 , 'host':host\_1} ,

            }

        print(self.ip\_blocked)

    #####################################

    ######### BandWidth Threads #########

    def bandwidth\_sample\_worker(self):

        worker = Worker(self.start\_calculate\_bandwitdh\_sample)

        #worker.signals.result.connect(self.ping\_checker\_results)

        self.threadpool.start(worker)

    def start\_calculate\_bandwitdh\_sample(self):

        """

            BandWidth Calculate Functions Start Thread

        """

        strt\_bndwds = "Start Bandwidth Thread"

        print(strt\_bndwds)

        io = psutil.net\_io\_counters()

        #print(io)

        update\_delay = 1 # in seconds

        # extract the total bytes sent and received

        bytes\_sent, bytes\_recv = io.bytes\_sent, io.bytes\_recv

        gui\_step\_by = self.findChild(qtw.QDoubleSpinBox,"doubleSpinBox\_bandwidth").text()

        gui\_timeout\_ping = self.findChild(qtw.QSpinBox,"spinBox\_timeout\_bandwidth").text()

        timeout\_counter = 0

        step\_by = float(gui\_step\_by)

        timeout\_ping = float(gui\_timeout\_ping)

        time\_st = datetime.now()

        print("Bandwidth Traffic Start")

        while timeout\_counter < timeout\_ping:

            res = calc\_bandwidth(self,bytes\_sent=bytes\_sent,bytes\_recv=bytes\_recv,update\_delay=update\_delay)

            QThread.msleep(step\_by)

            print(res)

            self.bandwidth\_set\_to\_dict(response\_dic=res)

            #timeout\_counter += step\_by

            time\_end = datetime.now()

            all\_time = time\_end - time\_st

            timeout\_counter = float(all\_time.seconds)

            print(all\_time.seconds)

    def bandwidth\_services\_worker(self):

        worker = Worker(self.start\_calculate\_bandwitdh\_services\_method1)

        worker2 = Worker(self.start\_calculate\_bandwitdh\_services\_method2)

        worker3 = Worker(self.start\_sniff\_services)

        worker.signals.result.connect(self.bandwitdh\_services\_results)

        self.threadpool.start(worker)

        global worker\_bandwidth\_1,worker\_bandwidth\_2 , worker\_bandwidth\_3

        worker\_bandwidth\_1 = worker

        worker\_bandwidth\_2 = worker2

        worker\_bandwidth\_3 = worker3

        self.threadpool.start(worker2)

        self.threadpool.start(worker3)

    def start\_calculate\_bandwitdh\_services\_method1(self):

        """

            Simple function that keeps printing the stats

        """

        strt\_bndwds = "Start Bandwidth Services Collect"

        print(strt\_bndwds)

        # extract the total bytes sent and received

        gui\_step\_by = self.findChild(qtw.QDoubleSpinBox,"doubleSpinBox\_bandwidth\_services").text()

        gui\_timeout\_ping = self.findChild(qtw.QSpinBox,"spinBox\_timeout\_bandwidth\_services").text()

        timeout\_counter = 0

        step\_by = float(gui\_step\_by)

        timeout\_ping = float(gui\_timeout\_ping)

        time\_st = datetime.now()

        print("Bandwidth Traffic Start")

        # self.tableWidget\_bandwidth\_services.setColumnCount(4)

        # columns = ('PID','Name','Download','Upload')

        # self.tableWidget\_bandwidth\_services.setHorizontalHeaderLabels(columns)

        while timeout\_counter < timeout\_ping and self.LOOPSRVS !=False:

            try:

                # self.bandwidth\_set\_to\_dict(response\_dic=res)

                #timeout\_counter += step\_by

                time\_end = datetime.now()

                all\_time = time\_end - time\_st

                all\_time = all\_time.seconds

                timeout\_counter = float(all\_time)

                response\_services = print\_pid2traffic()

                self.set\_services\_bandwidth\_to\_table(response\_services)

                time.sleep(step\_by)

            except Exception as err:

                time.sleep(step\_by)

                print("Error Bandwidth Traffic",err)

        print("Finished Bandwidth Traffic Start")

        self.LOOPSRVS = False

        #self.threadpool.clear()

        # self.threadpool.cancel(worker\_bandwidth\_1)

        self.threadpool.cancel(worker\_bandwidth\_2)

        self.threadpool.cancel(worker\_bandwidth\_3)

    def set\_services\_bandwidth\_to\_table(self,resp):

        """

            Convert From Global DataFrame which created by pandas to Gui Table Widget

        """

        self.tableWidget\_bandwidth\_services.clear()

        self.tableWidget\_bandwidth\_services.setRowCount(len(resp))

        dict\_resp = resp.to\_dict()

        #print(dict\_resp)

        if 'name' in dict\_resp:

            for key , dict\_values in dict\_resp.items():

                #print("key >>>   ",key)

                #print("dict\_values >>>>     ",dict\_values)

                count\_services = 0

                for pid,value in dict\_values.items():

                    #print("pid #",pid)

                    item\_pid = qtw.QTableWidgetItem(str(pid))

                    self.tableWidget\_bandwidth\_services.setItem(count\_services,0,item\_pid)

                    if key == "name":

                        item\_name = qtw.QTableWidgetItem(str(value))

                        self.tableWidget\_bandwidth\_services.setItem(count\_services,1,item\_name)

                    if key == "Download":

                        item\_download = qtw.QTableWidgetItem(str(value))

                        self.tableWidget\_bandwidth\_services.setItem(count\_services,2,item\_download)

                    if key == "Upload":

                        item\_upload = qtw.QTableWidgetItem(str(value))

                        self.tableWidget\_bandwidth\_services.setItem(count\_services,3,item\_upload)

                        #print("Key", key)

                        #print("value ############ ",value)

                    #print("dict\_resp",dict\_resp['name'])

                    count\_services += 1

    def bandwitdh\_services\_results(self):

        QThread.sleep(2)

        self.LOOPSRVS = True

        print("Timer Bandwidth Services Finished")

    def start\_calculate\_bandwitdh\_services\_method2(self):

        """

            A function that keeps listening for connections on this machine

            and adds them to `connection2pid` global variable

        """

        global connection2pid

        gui\_step\_by = self.findChild(qtw.QDoubleSpinBox,"doubleSpinBox\_bandwidth\_services").text()

        gui\_timeout\_ping = self.findChild(qtw.QSpinBox,"spinBox\_timeout\_bandwidth\_services").text()

        timeout\_counter = 0

        step\_by = float(gui\_step\_by)

        timeout\_ping = float(gui\_timeout\_ping)

        time\_st = datetime.now()

        #print("Bandwidth Traffic Start")

        while self.LOOPSRVS != False:

            try:

                time\_end = datetime.now()

                all\_time = time\_end - time\_st

                all\_time = all\_time.seconds

                timeout\_counter = float(all\_time)

                #while True:

                    # using psutil, we can grab each connection's source and destination ports

                    # and their process ID

                for c in psutil.net\_connections():

                    if c.laddr and c.raddr and c.pid:

                        # if local address, remote address and PID are in the connection

                        # add them to our global dictionary

                        connection2pid[(c.laddr.port, c.raddr.port)] = c.pid

                        connection2pid[(c.raddr.port, c.laddr.port)] = c.pid

                #time.sleep(step\_by)

                    #print(connection2pid)

            except Exception as err:

                print("Error",err)

    def start\_sniff\_services(self):

        gui\_step\_by = self.findChild(qtw.QDoubleSpinBox,"doubleSpinBox\_bandwidth\_services").text()

        gui\_timeout\_ping = self.findChild(qtw.QSpinBox,"spinBox\_timeout\_bandwidth\_services").text()

        timeout\_counter = 0

        step\_by = float(gui\_step\_by)

        timeout\_ping = float(gui\_timeout\_ping)

        time\_st = datetime.now()

        while self.LOOPSRVS != False:

            try:

                time\_end = datetime.now()

                all\_time = time\_end - time\_st

                all\_time = all\_time.seconds

                timeout\_counter = float(all\_time)

                sc.sniff(prn=process\_packet, store=False)

                #time.sleep(step\_by)

            except Exception as err:

                print("Error",err)

                #time.sleep(step\_by)

    def process\_packet(self,packet):

        global pid2traffic

        try:

            # get the packet source & destination IP addresses and ports

            packet\_connection = (packet.sport, packet.dport)

        except (AttributeError, IndexError):

            # sometimes the packet does not have TCP/UDP layers, we just ignore these packets

            pass

        else:

            # get the PID responsible for this connection from our `connection2pid` global dictionary

            packet\_pid = connection2pid.get(packet\_connection)

            if packet\_pid:

                if packet.src in all\_macs:

                    # the source MAC address of the packet is our MAC address

                    # so it's an outgoing packet, meaning it's upload

                    pid2traffic[packet\_pid][0] += len(packet)

                else:

                    # incoming packet, download

                    pid2traffic[packet\_pid][1] += len(packet)

    def bandwidth\_set\_to\_dict(self,response\_dic):

        """

            Gui Functions Signal To Set Results Bandwidth

        """

        res = response\_dic

        self.label\_bandwidth\_down\_speed.setText(res['Download Speed'])

        self.label\_bandwidth\_upload\_speed.setText(res['Upload Speed'])

        self.lineEdit\_bandwidth\_down.setText(res['Download'])

        self.lineEdit\_bandwidth\_upload.setText(res['Upload'])

    ###################################

    ########### Message Box ###########

    ###################################

    def message\_error(self, s,timer\_sleep=None):

        if timer\_sleep == None:

            print("Error", s)

            dlg = QMessageBox(self)

            dlg.setText(s)

            dlg.setIcon(QMessageBox.Warning)

            dlg.setStandardButtons(QMessageBox.Ok)

            dlg.setWindowTitle("Error")

            button = dlg.exec\_()

        else:

            print("Error Timer", s)

            dlg = QMessageBox(self)

            dlg.setText(s)

            dlg.setWindowTitle("Error")

            button = dlg.exec\_()

    ###################################

    ########### Start Ping ############

    ###################################

    def thread\_ping\_all(self):

        worker = Worker(self.start\_ping\_all\_scanned)

        worker.signals.result.connect(self.ping\_all\_scanned\_result)

        self.threadpool.start(worker)

    def thread\_ping\_check\_auto\_save(self):

        worker = Worker(self.ping\_checker\_auto)

        #worker.signals.result.connect(self.ping\_checker\_results)

        self.threadpool.start(worker)

    def start\_ping\_all\_scanned(self):

        print("Ping All Scanned")

        gui\_step\_by = self.findChild(qtw.QSpinBox,"spinBox\_step\_ping\_all").text()

        auto\_check = self.checkBox\_ping\_timer\_all.isChecked()

        ping\_save\_check = self.checkBox\_ping\_save\_db\_home.isChecked()

        listen\_dev\_check = self.checkBox\_listen\_dev\_off.isChecked()

        dev\_save\_check = self.checkBox\_save\_dev\_db.isChecked()

        step\_by = float(gui\_step\_by)

        print(auto\_check)

        if auto\_check:

            while self.LOOPPINGER:

                try:

                    online\_devices = 0

                    self.pushButton\_ping\_all.setText("Pinging ...")

                    self.pushButton\_ping\_all\_stop.setEnabled(True)

                    self.pushButton\_ping\_all.setEnabled(False)

                    for row\_num , row\_info in ip\_scanned.items():

                        try:

                            check\_ip = self.findChild(qtw.QCheckBox,f"checkBox\_ping\_{str(row\_num)}").isChecked()

                        except:

                            continue

                        if check\_ip == True:

                            resp = self.which\_ip\_to\_ping(num\_ip=row\_num)

                            if resp['Response'] == True:

                                online\_devices += 1

                            else:

                                if listen\_dev\_check == True:

                                    self.error\_message = f"Device With Ip {resp['IP']} and Mac Address {resp['Mac']} Has an error and sent to administration e-mail"

                                    self.LOOPPINGER = False

                                    break

                            if dev\_save\_check == True:

                                try:

                                    print("Start To Create Device")

                                    ip = resp['IP']

                                    mac = resp['Mac']

                                    self.db.connect()

                                    try:

                                        dev = Device.select().where(Device.mac\_address == mac).get()

                                        print(dev)

                                    except:

                                        dev = None

                                    if dev != None:

                                        print("Found",dev)

                                        self.db.close()

                                    else:

                                        print(ip)

                                        print(mac)

                                        #self.db.connect()

                                        dev = Device.create(

                                            ip\_dev=ip,

                                            mac\_address=mac,

                                            )

                                        dev.save()

                                        self.db.close()

                                        print("Device Created",ip,"Success")

                                except "UNIQUE" in Exception:

                                    print("Device Already Inserted : ",err)

                                    self.db.close()

                                    pass

                                except Exception as err:

                                    print("Failed To Create Instance : ",err)

                                    self.db.close()

                                    pass

                            if ping\_save\_check == True:

                                try:

                                    print("Start To Create Ping")

                                    self.db.connect()

                                    dev = Device.select().where(Device.mac\_address == resp['Mac']).get()

                                    print(dev)

                                    #print(dev.ip\_dev)

                                    ping\_obj = PingInfo.create(

                                        owner=dev,

                                        is\_anwsred = resp['Response'],

                                        resp\_time = resp['ResTime'],

                                        ttl = resp['TTL'],

                                    )

                                    ping\_obj.save()

                                    self.db.close()

                                    print("Created","IP",resp['IP'],"Response Time",resp['ResTime'])

                                except Exception as err:

                                    self.db.close()

                                    print("Error In Save To db > ",err)

                    print("Online Devices",online\_devices)

                    self.lcdNumber.display(online\_devices)

                    time.sleep(step\_by)

                except NameError:

                    print("Scan Network First Please ....")

                    self.LOOPPINGER = False

                except Exception as err:

                    print("Error",err)

                    self.LOOPPINGER = False

        else:

            try:

                for row\_num , row\_info in ip\_scanned.items():

                    check\_ip = self.findChild(qtw.QCheckBox,f"checkBox\_ping\_{str(row\_num)}").isChecked()

                    if check\_ip == True:

                        self.which\_ip\_to\_ping(num\_ip=row\_num)

                time.sleep(step\_by)

            except NameError:

                print("Scan Network First Please ....")

                self.LOOPPINGER = False

            except Exception as err:

                print("Error",err)

                self.LOOPPINGER = False

    def ping\_all\_scanned\_result(self):

        print("Ping All Results ")

        try:

            if len(ip\_scanned) > 0:

                print("Scanned")

            else:

                print("Not Scanned")

        except NameError:

            self.message\_error(s="Scan Network First Please")

        if self.error\_message != None:

            self.message\_error(s=str(self.error\_message))

            self.error\_message = None

        btn\_stop\_ping = self.findChild(qtw.QPushButton,"pushButton\_ping\_all\_stop")

        btn\_strt\_ping = self.findChild(qtw.QPushButton,"pushButton\_ping\_all")

        btn\_strt\_ping.setText("Start Ping")

        btn\_stop\_ping.setEnabled(False)

        btn\_strt\_ping.setEnabled(True)

        self.LOOPPINGER = True

    def stop\_ping(self):

        print("Ping Stopped")

        self.LOOPPINGER = False

    def ping\_checker\_auto(self,main=None):

        #time\_st = datetime.datetime.now().second

        print(main)

        gui\_step\_by = self.findChild(qtw.QDoubleSpinBox,"doubleSpinBox\_ping\_stepby").text()

        gui\_timeout\_ping = self.findChild(qtw.QSpinBox,"spinBox\_ping\_timeout").text()

        timeout\_counter = 0

        step\_by = float(gui\_step\_by)

        timeout\_ping = float(gui\_timeout\_ping)

        time\_st = datetime.now()

        ip\_scanned\_checked = self.checkBox\_ping\_ip\_scanned.isChecked()

        multi\_check = self.checkBox\_ping\_timer.isChecked()

        save\_device\_db\_check = self.checkBox\_save\_device\_db.isChecked()

        if main == True:

            ip\_scanned\_checked = True

            gui\_step\_by = self.findChild(qtw.QDoubleSpinBox,"spinBox\_step\_ping\_all").text()

            step\_by = float(gui\_step\_by)

        print("Ping Checker Is Running")

        if save\_device\_db\_check:

            list\_to\_check = []

            for n in range(4):

                try:

                    text = self.findChild(qtw.QLineEdit,f'lineEdit\_ip\_to\_ping\_{str(n)}').text()

                    if text != "":

                        list\_to\_check.append(text)

                except:

                    pass

            if ip\_scanned\_checked != True:

                print("IP Scanned Manual",list\_to\_check)

                if len(list\_to\_check) >= 1:

                    for ip in list\_to\_check:

                        try:

                            #print("Create IP Scanned In Database Info")

                            self.insert\_ip\_to\_db(scanned\_dic=False,ip\_to\_save=ip)

                        except:

                            self.db.close()

                            pass

            else:

                try:

                    #print("Create IP Scanned In Database Info")

                    self.insert\_ip\_to\_db()

                except:

                    self.db.close()

                    pass

        if multi\_check:

            while timeout\_counter < timeout\_ping:

                self.ping\_checker()

                QThread.sleep(step\_by)

                #timeout\_counter += step\_by

                time\_end = datetime.now()

                all\_time = time\_end - time\_st

                timeout\_counter = float(all\_time.seconds)

                print(all\_time.seconds)

        else:

            self.ping\_checker()

        #print(str\_time)

    def thread\_ping\_save\_in\_db(self):

        worker = Worker(self.ping\_checker)

        #worker.signals.result.connect(self.ping\_checker\_results)

        self.threadpool.start(worker)

    def ping\_checker(self,from\_main=False):

        ip\_scanned\_checked = self.checkBox\_ping\_ip\_scanned.isChecked()

        save\_ping\_to\_db\_checked = self.checkBox\_save\_ping\_db.isChecked()

        save\_device\_to\_db\_checked = self.checkBox\_save\_device\_db.isChecked()

        list\_to\_check = []

        for n in range(4):

            try:

                text = self.findChild(qtw.QLineEdit,f'lineEdit\_ip\_to\_ping\_{str(n)}').text()

                if text != "":

                    list\_to\_check.append(text)

            except:

                pass

        #print("Start Create If Not In DB")

        #print("Start Ping Checker")

        if ip\_scanned\_checked:

            try:

                for row\_num , row\_info in ip\_scanned.items():

                    ip = row\_info['IP']

                    mac = row\_info['Mac']

                    ## Get Ip Ping Response And prepare it to save in db if checked

                    ip\_ping\_response = self.collect\_ping\_to\_dict(ip)

                    if save\_ping\_to\_db\_checked:

                        try:

                            print("Start To Create Ping")

                            self.db.connect()

                            dev = Device.select().where(Device.mac\_address == mac).get()

                            #print(dev)

                            #print(dev.ip\_dev)

                            ping\_obj = PingInfo.create(

                                owner=dev,

                                is\_anwsred = ip\_ping\_response['Response'] ,

                                details = ip\_ping\_response

                            )

                            ping\_obj.save()

                            self.db.close()

                            print("Created",ip)

                        except Exception as err:

                            self.db.close()

                            print("Error In Save To db > ",err)

            except NameError:

                print("Scan Network First Please ....")

            except Exception as err:

                print("Error",err)

                pass

        else:

            for ip in list\_to\_check:

                ip\_ping\_response = self.collect\_ping\_to\_dict(ip)

                print(ip\_ping\_response)

                if save\_ping\_to\_db\_checked:

                    try:

                        self.db.connect()

                        ping\_obj = PingInfo.create(

                                is\_anwsred = ip\_ping\_response['Response'] ,

                                details = ip\_ping\_response

                            )

                        ping\_obj.save()

                        self.db.close()

                        print("Created",ip)

                    except Exception as err:

                        self.db.close()

                        print("Error",err)

                        pass

            #print(list\_to\_check)

    def insert\_ip\_to\_db(self,scanned\_dic=True,ip\_to\_save=None):

        try:

            if scanned\_dic == True:

                print(ip\_scanned)

                print(len(ip\_scanned))

                for row\_num , row\_info in ip\_scanned.items():

                    ip = row\_info['IP']

                    mac = row\_info['Mac']

                    try:

                        self.db.connect()

                        ip\_1 = Device.create(

                            ip\_dev=ip,

                            mac\_address=mac

                            )

                        ip\_1.save()

                        self.db.close()

                        print("Device Created", ip,mac ,"Success")

                    except "UNIQUE" in Exception:

                        print("Device Already Inserted : ",err)

                        self.db.close()

                    except Exception as err:

                        print("Failed To Create Instance : ",err)

                        self.db.close()

            else:

                print(ip\_to\_save)

                try:

                    self.db.connect()

                    ip\_1 = Device.create(

                        ip\_dev=ip\_to\_save,

                        is\_local=False,

                        )

                    ip\_1.save()

                    self.db.close()

                    print("Device Created", ip\_to\_save ,"Success")

                except "UNIQUE" in Exception:

                    print("Device Already Inserted : ",err)

                    self.db.close()

                except Exception as err:

                    print("Failed To Create Instance : ",err)

                    self.db.close()

        except NameError:

            print("Scan Network First Please ....")

            self.db.close()

            pass

        except:

            self.db.close()

            pass

    def collect\_ping\_to\_dict(self,ip\_to\_ping):

        ip\_ping\_response = {

            'IP':ip\_to\_ping,

            'Mac':None,

            'Checked':False,

            'Response':False,

            'TTL':None,

            'ResTime':None,

        }

        if ip\_to\_ping == "" :

            pass

        else:

            ping\_response = self.ping\_ip(ip\_to\_ping)

            if "Reply from" in ping\_response and \

                not "Destination host unreachable." in ping\_response:

                ## Find Results In Respons String In Terminal

                ttl\_index = ping\_response.find("TTL=")

                ttl = ping\_response[ttl\_index+4:ttl\_index+7].strip()

                time\_index = ping\_response.find("time=")

                time\_index\_end = ping\_response.find("ms TTL")

                time\_ping = ping\_response[time\_index+5:time\_index\_end]

                ## Set Results To variable ip\_ping\_response type(dict)

                ip\_ping\_response['Checked'] = True

                ip\_ping\_response['Response'] = True

                ip\_ping\_response['TTL'] = ttl

                ip\_ping\_response['ResTime'] = time\_ping

            if  "Reply from" in ping\_response :

                ip\_ping\_response['Checked'] = True

                print(ping\_response)

                if "Destination host unreachable." in ping\_response:

                    ip\_ping\_response['Response'] = False

        return ip\_ping\_response

    def ping\_checker\_results(self):

        print("Done Results")

    ### Table Widget Ping And Device ###

    def set\_table\_widget\_from\_db(self):

        worker = Worker(self.get\_table\_rows)

        worker.signals.result.connect(self.results\_get\_info\_table)

        self.threadpool.start(worker)

    def get\_table\_rows(self):

        from db\_to\_qt import devss

        from db\_to\_qt import pingss

        global dev\_db , ping\_db , max\_ping\_table

        dev\_db = devss

        ping\_db = pingss

        max\_ping\_table = 100

        self.tableWidget1\_ping.setRowCount(len(ping\_db))

        # self.tableWidget1\_ping.setRowCount(max\_ping\_table)

        print(len(devss))

        print(len(pingss))

    def results\_get\_info\_table(self):

        print("Done Get Data")

        try:

            for counter , ping in enumerate(ping\_db):

                try:

                    time\_ping = str(ping.created\_at).split(".")[0]

                except:

                    pass

                try:

                    owner\_ip = str(ping.owner.ip\_dev)

                except:

                    owner\_ip = "Not Found"

                item\_id = qtw.QTableWidgetItem(str(ping.id))

                item\_anwser = qtw.QTableWidgetItem(str(ping.is\_anwsred))

                item\_time = qtw.QTableWidgetItem(str(time\_ping))

                item\_owner = qtw.QTableWidgetItem(str(owner\_ip))

                self.tableWidget1\_ping.setItem(counter,1,item\_owner)

                #item\_det = qtw.QTableWidgetItem(str(ping.details))

                #self.tableWidget1\_ping.setItem(counter,4,item\_det)

                self.tableWidget1\_ping.setItem(counter,0,item\_id)

                self.tableWidget1\_ping.setItem(counter,2,item\_anwser)

                self.tableWidget1\_ping.setItem(counter,3,item\_time)

        except Exception as err:

            print(err)

        print("Finished")

def create\_app():

    #widget\_meters = AnalogGaugeWidget()

    #window = Main()

    window = Login()

    window.show()

    app.exec\_()

if \_\_name\_\_ == '\_\_main\_\_':

    app = qtw.QApplication(sys.argv)

    app.setStyle('Fusion')

    #engine = QQmlApplicationEngine()

    #engine.load(os.path.join(os.path.dirname(\_\_file\_\_), "qml/main.qml"))

    # with open("front/style.qss", "r") as f:

    #     \_style = f.read()

    #     app.setStyleSheet(\_style)

    list\_theme = [

        'dark\_amber.xml',

        'dark\_blue.xml',

        'dark\_cyan.xml',

        'dark\_lightgreen.xml',

        'dark\_pink.xml',

        'dark\_purple.xml',

        'dark\_red.xml',

        'dark\_teal.xml',

        'dark\_yellow.xml',

        'light\_amber.xml',

        'light\_blue.xml',

        'light\_cyan.xml',

        'light\_cyan\_500.xml',

        'light\_lightgreen.xml',

        'light\_pink.xml',

        'light\_purple.xml',

        'light\_red.xml',

        'light\_teal.xml',

        'light\_yellow.xml'

    ]

    #apply\_stylesheet(app, theme=list\_theme[1])

    app.setStyleSheet(qdarkstyle.load\_stylesheet\_pyside2())

    create\_app()

    is\_program\_running = True