**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

Jnana Sangam, Belagavi-590018



### Project Report

on

# ENHANCING NETWORKING MONITORING SYSTEMS BY OVERLAYING PROTOCOLS

by

**RAJASHREE**

**1VI17CS114**

**PAVITHRA K**

**1VI17CS071**

UNDER THE GUIDANCE OF

### Mr. NOOR PASHA

Asst. Professor Department of CSE, Vemana Institute of Technology



### DEPARTMENT OF COMPUTER SCIENCE ENGINEERING,

### VEMANA INSTITUTE OF TECHNOLOGY

3RD BLOCK, KORAMANGALA, BENGALURU – 560034

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **Acknowledgement** | **i** |
| **Abstract** |  |
| **List of Figures** |  |
| **List of Tables** |  |
| 1. **Introduction** | **1** |
| * 1. **Scope** | **1** |
| * 1. **Objective** | **2** |
| 1. **Literature Survey** |  |
| * 1. **Comparative Analysis** |  |
| 1. **System Requirements** |  |
| * 1. **Functional Requirements** |  |
| * 1. **Non-Functional Requirements** |  |
| * 1. **Hardware Requirements** |  |
| * 1. **Software Requirements** |  |
| 1. **Design Methodology** |  |
| * 1. **System Architecture** |  |
| * 1. **Design** |  |
| * + 1. **Use Case Diagram** |  |
| * + 1. **Class Diagram** |  |
| * + 1. **Sequence Diagram** |  |
| * + 1. **Data Flow Diagram** |  |
| 1. **Applications** |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 Scope**

A computer network is a group of network nodes that use a set of common communication protocols over digital interconnections for the purpose of sharing resources located on or provided by the network nodes. A network node is either a redistribution point or a communication endpoint such as modem, hub, switch, bridge, routers, network file systems, personal computers and so on. Network monitoring system is a scripted tool that administers a computer network for slow or failing components and that notifies the configured users in case of outages, latencies, upgrades, and other events.

With the advent of internet, network monitors can now perform variating services such as fault analysis, performance management, provisioning of networks and maintain quality of service. Internet technologies have become very important in every person’s day to day life and proactive monitoring of interconnected devices has become vital in the internet service provision business. The onset of ecommerce has reduced the prices of IoT to great extent and a typical home in suburban consists of Layer 3 router, a Wi-Fi access point, internet connected TV and remotely controlled light bulbs. As a result, security and monitoring have become a critical concern for every person with internet connectivity.

The latest generation of network monitoring systems are designed to support very specific applications. Most of them in general, are devised to monitor enterprise environments that generally consist of 100s of similar network nodes such as a network rack, routers, switches, and personal computers. Further, they are designed to support very specific applications such as identifying the device’s connectivity or latency or out-of-band analysis and so on. To obtain results they are homogenous in terms of protocol usage. For example, some systems use ICMP alone to obtain performance analysis of a huge network. Also, due to the lack of a buffer Dataset, they are not generally real-time and are quite torpid. A typical topology that consists of less than 50 network nodes such as a startup environment or a home office environment cannot afford such a stack of tools and maintaining a stable and high performance network is an up-hill task for them. As a result, most of the emerging organizations employ a separate team that act as network administrators.

**1.2 Objective**

**CHAPTER 2**

**LITERATURE SURVEY**

The survey of various papers on Network Monitors is presented in the following section:

**[1]** **A transparent virtual machine monitor level packet compression network service – by Ali Hamidi, Hadi Salimi and Mohsen Sharifi. [2010]:**

This paper explains about packet compression, a ubiquitous technique for improving the performance of low speed networks such as WANs, especially effective in networks with high cost per transmitted byte such as wireless networks. It also shows how packer compression service can be used in virtualization technology in which network packet payloads are compressed and hence application communication overhead involved in virtual machines is reduced. The feasibility of development and dynamic configuration of such service and the extensive improvement of network performance over congested links is proven in the results and provided as a proof of concept.

**[2] A Monitor Tool for a Network Based on the Cambridge Ring – by Synnove Vassiliades, Michael D. Sayers and Jean M. Bacon. [1986]:**

This paper shows how networks and their demands have evolved over time and different options to monitor a network. Also, it illustrates how to validate analytic and simulate models for network topologies, detect implementation errors, performance bugs and so on. It also describes a measurement facility which has been developed to discuss the issues and practical constraints involved in its design to suggest enhancements in the hatfield network.

**[3] Design and implementation of a Web-based Internet Performance Management System Using SNMP MIB-II – by Seong Jin Ahn, Seung Keun yoo and Jin Wook Chung [1999]:**

This paper analyses various attributes required for monitoring performance of a network and how to extract them using the SNMP MIB-II and MATP protocol. It uses Java to form a web-based tool to show the current performance of a network. Although it was first of its kind, this model lacks a buffer databased between the graphical user interface and back end implementation which makes the data loading slow, especially in virtual private networks.

**[4] Network Management in the work of standards: The Role of the SNMP Protocol in Managing Networks – by Dr Katherine Jones [1991]:**

This white paper gives a high level view of the SNMP protocol, current standards in network management, MIBs, responsibilities of a SNMP manager, examples of SNMP Implementations, issues with current GUIs in SNMP, selection criteria for purchasing a network manager and various vendors. This paper was particularly useful in articulating the current proposed system.

**[5] On evaluating the differences of TCP and ICMP in network measurement – by Li Wenwei, Zhange Dafang, Yang Jinmin and Xie Gaogang [2007]:**

This paper the parameters and results of measuring host connectivity, RTT andpacket loss rate are compared between TCP and ICMP. While the accuracy of the results are higher in TCP, the time taken was significantly less in ICMP especially in the case of calculating RTT.

**2.1 Comparative Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref** | **Technique** | **Advantage** | **Drawback** |
| [1] | QoS and VoIP | Balanced Cost and Quality | Localized to large enterprises |
| [2] | Multi-Cloud | Flexible and first of its kind | Network intensive and simulated results |
| [3] | Routing | Method to implement a framework | No Real-time analysis |
| [4] | VPN | Integration of GUI with back end | Covers only VPNs |

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**3.1 Functional Requirements**

1. To instantly detect device outages or performance reductions and immediately trigger notifications via E-Mail/SMS/GUI alerts to configured users.
2. To perform polling of device statistics in real-time to measure various performance attributes such as reachability, availability, bandwidth, and latency.
3. To provide easy to view and comprehensive web-based application user interface that automatically obtains device info, uptime, real-time statistics, traffic statistics and so on.
4. To provide periodic historical reports of devices on their performance.
5. To facilitate accessing different network nodes via SSH/Telnet/Web access from the monitor itself.
6. To ate custom configuration of notifications to different types of users in various situations through various mediums.
7. To add support monitoring of various network nodes such as Servers, Routers, Switches, Power Distribution Units (Network PDUs), Virtual Machines, IoT devices, Cloud instances, Data Stores, Wireless Access Points, Endpoint PCs and so on.

**3.2 Non-Functional Requirements**

* Reliability: The system should inform user when it operates in degraded mode. It should be able to inform user about any malfunction and be able to handle failures. System should be able to deliver the required service in reliable manner, i.e. for the input or state of the system the output obtained must be correct. Reliability is an attribute of any computer-related component (software, or hardware, or a network, for example) that consistently performs according to its specifications. It has long been considered one of three related attributes that must be considered when making, buying, or using a computer product or component. Reliability, availability, and serviceability - RAS, for short - are important aspects to design into any system. In theory, a reliable product is totally free of technical errors; in practice, however, vendors frequently express a product's reliability quotient as a percentage. Evolutionary products (those that have evolved through numerous versions over a significant period of time) are usually considered to become increasingly reliable, since it is assumed that bug s have been eliminated in earlier releases. For example, IBM's z/OS (an operating system for their S/390 server series), has a reputation for reliability because it evolved from a long line of earlier MVS and OS/390 operating system versions.
* Availability: Availability refers to the frequency at which the service provided by the system is available, if the system can be accessed at any point of time then that system is highly available. Availability of a system is typically measured as a factor of its reliability - as reliability increases, so does availability. Availability of a system may also be increased by the strategy on focusing on increasing testability & maintainability and not on reliability. Improving maintainability is generally easier than reliability. Maintainability estimates (Repair rates) are also generally more accurate. However, because the uncertainties in the reliability estimates are in most cases very large, it is likely to dominate the availability (prediction uncertainty) problem, even while maintainability levels are very high. When reliability is not under control more complicated issues may arise, like manpower (maintainers / customer service capability) shortage, spare part availability, logistic delays, lack of repair facilities, extensive retrofit and complex configuration management costs and others. The problem of unreliability may be increased also due to the "domino effect" of maintenance induced failures after repairs. Only focusing on maintainability is therefore not enough. If failures are prevented, none of the others are of any importance and therefore reliability is generally regarded as the most important part of availability. Reliability needs to be evaluated and improved related to both availability and the cost of ownership (due to cost of spare parts, maintenance man-hours, transport costs, storage cost, part obsolete risks etc.). Often a trade-off is needed between the two. There might be a maximum ratio between availability and cost of ownership. Testability of a system should also be addressed in the availability plan as this is the link between reliability and maintainability. The maintenance strategy can influence the reliability of a system (e.g. by preventive and/or predictive maintenance), although it can never bring it above the inherent reliability. So, Maintainability and Maintenance strategies influences the availability of a system. In theory this can be almost unlimited if one would be able to always repair any fault in an infinitely short time. This is in practice impossible. Repair-ability is always limited due to testability, manpower and logistic considerations.
* Extensibility: The system should be extensible to add further information and users for more expansion. In software engineering, extensibility (not to be confused with forward compatibility) is a system design principle where the implementation takes future growth into consideration. It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension. Extensions can be through the addition of new functionality or through modification of existing functionality. The central theme is to provide for change, typically enhancements, while minimizing impact to existing system functions.

Extensibility is a software design principle defined as a system’s ability to have new functionality extended, in which the system’s internal structure and data flow are minimally or not affected, particularly that recompiling or changing the original source code is unnecessary when changing a system’s behaviour, either by the creator or other programmers. Because software systems are long lived and will be modified for new features and added functionalities demanded by users, extensibility enables developers to expand or add to the software’s capabilities and facilitates systematic reuse. Some of its approaches include facilities for allowing users’ own program routines to be inserted and the abilities to define new data types as well as to define new formatting mark-up tags.

* Re-Usability: The system’s code could be reused to add further new features if need to be added in future. In computer science and software engineering, reusability is the use of existing assets in some form within the software product development process. More than just code, assets are products and by-products of the software development life cycle and include software components, test suites, designs and documentation. Leverage is modifying existing assets as needed to meet specific system requirements. Because reuse implies the creation of a separately maintained version of the assets, it is preferred over leverage. Subroutines or functions are the simplest form of reuse. A chunk of code is regularly organized using modules or namespaces into layers. Proponents claim that objects and software components offer a more advanced form of reusability, although it has been tough to objectively measure and define levels or scores of reusability. The ability to reuse relies in an essential way on the ability to build larger things from smaller parts, and being able to identify commonalities among those parts. Reusability is often a required characteristic of platform software. Reusability brings several aspects to software development that do not need to be considered when reusability is not required. Reusability implies some explicit management of build, packaging, distribution, installation, configuration, deployment, and maintenance and upgrade issues. If these issues are not considered, software may appear to be reusable from design point of view, but will not be reused in practice. Software reusability more specifically refers to design features of a software element (or collection of software elements) that enhance its suitability for reuse.
* Robustness: The system must be tolerant enough to cope up with minor errors during it run time. In computer science, robustness is the ability of a computer system to cope with errors during execution. Robustness can also be defined as the ability of an algorithm to continue operating despite abnormalities in input, calculations, etc. Robustness can encompass many areas of computer science, such as robust programming, robust machine learning, and Robust Security Network form, the more robust the software. Formal techniques, such as fuzz testing, are essential to showing robustness since this type of testing involves invalid or unexpected inputs. Alternatively, fault injection can be used to test robustness. Various commercial products perform robustness testing of software systems and is a process of failure assessment analysis. In general, building robust systems that encompass every point of possible failure is difficult because of the vast amount of possible inputs and input combinations. Since all inputs and input combinations would require too much time to test, developers cannot run through all cases exhaustively. Instead, the developer will try to generalize such cases. Interoperability: New updates to the system must be easily accommodated in future. Interaction with the user must be minimal. System is to be developed in phases, so it shall be easily upgradeable. Upgrading is the process of replacing a product with a newer version of the same product. In computing and consumer electronics an upgrade is generally a replacement of hardware, software or firmware with a newer or better version, in order to bring the system up to date or to improve its characteristics. Common software upgrades include changing the version of an operating system, of an office suite, of an anti-virus program, or of various other tools.
* Performance: The system should provide measures on its power consumption. Various parameters such as response time, heat dissipation is to be observed.
* Efficiency: Power optimization is a critical concern in the system. Accurate Communication on various states of the system to the user is important.
* Maintainability: Easy Learning of the system is necessary for the maintenance personnel to repair it easily in the events of failure.
* Safety and Security: Measures to prevent unauthorized access to Robotic Car and data in it are taken. Functionality that operates in a manner that does not encourage unsafe behaviour is provided.
* Scalability: The system must be able to scale itself at various levels of performance. Required resources of the system should increase proportionally.
* Usability: The system should be developed to be simple and efficient for the end users and easy to understand. It ensures consistency of interface and service.
* Durability: As the hardware components are moving along with the car, the hardware’s used must be tolerant to shocks and external noises. Durability is the ability of the system to endure.

**3.3 Hardware Requirements**

The following section describes the various hardware requirements for a network monitor to be developed and used:

* Processor: Intel® Core i5 ™ CPU and above
* RAM: 8 GB or higher
* Hard Disk: 100 GB or higher

**3.4 Software Requirements**

The following section describes the various hardware requirements for a network monitor to be developed and used:

* Operating System: Windows 10/Ubuntu 20.04 LTS
* Architecture: 64-bit OS
* Python 3.8 or higher
* PIP Packages: RegEx, Django, Pymysql
* Database: MySQL5.7 or higher
* JavaScript 1.8.5 or higher
* Front End: HTML5, CSS3, Bootstrap4

**CHAPTER 4**

**DESIGN METHODOLOGY**

A design document is a primary document of agreement between a customer and a software engineer. It consists of the various functional and non-functional requirements of software in a language that is understood both by the user and the software engineer. Design documents usually have various levels of details. The high-level design document describes the abstract or the outer functionalities of the system. The low-level design documents focus on the minute and internal details of the system.

**4.1 System Architecture:**

**Graphical User Interface:**

* This facilitates the user to login into the system, initiate the utility, whereby detail of the entire network is present in a table.
* This table is automatically filled with details of the network node such as MAC Address, OS and so on using LLDP protocol
* The Traffic Stats will show up eventually which are obtained from the device using ICMP protocol
* The user can also see all the performance attributes such as reachability, latency and so on present on any network node by switching between tabs.
* The user can take SSH/Web/Telnet access to the network node by clicking on Access button in the table.
* The user can update, modify, change, or configure the network nodes. This triggers a background process of history keeping. It also provides the ability to caution any change of an attribute which may not be compatible with the resource at hand.
* The GUI also shows latest alerts on the top with various color codes assigned based on the criticality of the event.
* Types of Alerts and their frequency can be configured under the Profile option.
* Coding Languages used for this Component: HTML, CSS, Bootstrap Templates, JavaScript

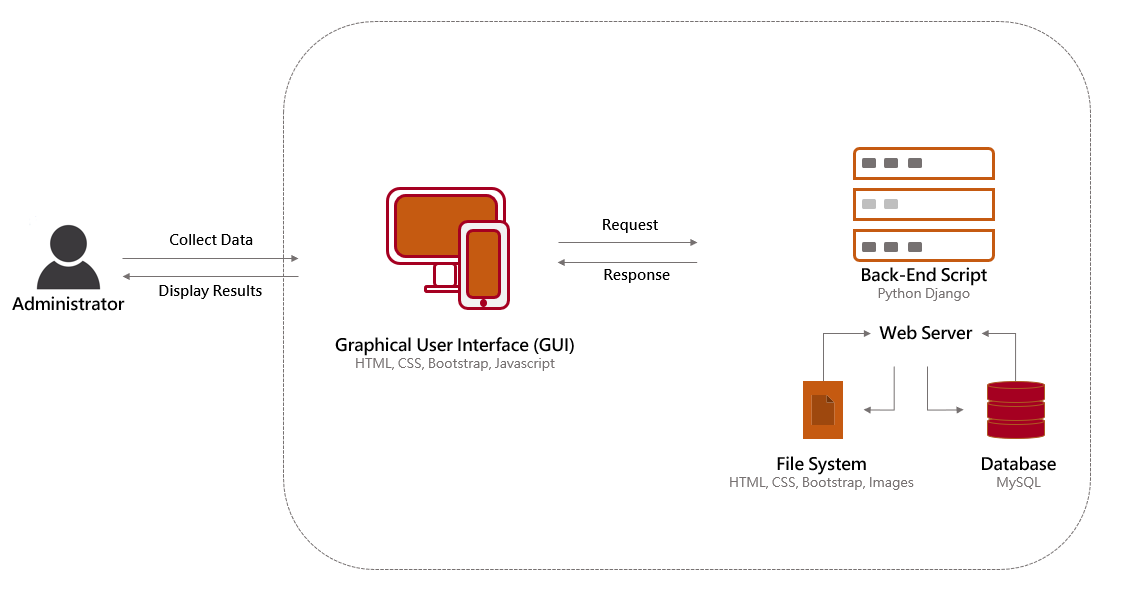
**Back-end Processing:**

* This module handles all functionality from finding the network nodes, attributes, querying with the network nodes and updating the database with information obtained through back end processing
* It also maintains a dynamically updated database which contains all the attributes present on every resource along with compatibility problems. As a result, it can take decisions on issues occurred and report errors accordingly.
* Various protocols are used to obtain different types of data for example LLDP for device information, SNMP for performance statistics, SSH/Telnet/HTTP for device access, ICMP for traffic statistics and so on.
* Raw IP sockets are formed and sent in various ways to determine the operating system the network nodes are using, packet filters in use and several other attributes.
* Coding Languages used for this Component: Python, Django/Flask Framework, JavaScript, pymysql framework

**Database:**

* This module handles acts a buffer between the backend functionality and the graphical user interface. It provides synchronization between the activities of the user and the backend
* To avoid top down triggering and querying of the network node data every time the web page is loaded or meta-refreshed, the backend functionality runs infinitely to fill various attributes of the network node by querying it using various network protocols at regular intervals.
* Simultaneously, GUI fetches data from database which results in real-time transmission at any given second.
* MySQL Tables used:
  + UserAuthentication: Deals with user login, logout, and session keys
  + Devices: Deals with devices and their attributes at real time
* Languages used for this Component: MySQL Database, MySQL query Language, MySQL Workbench / MySQL CLI for diagnostics

The system architecture of the proposed system is illustrated in the Figure 4.1



**Fig: 4.1 System Architecture**

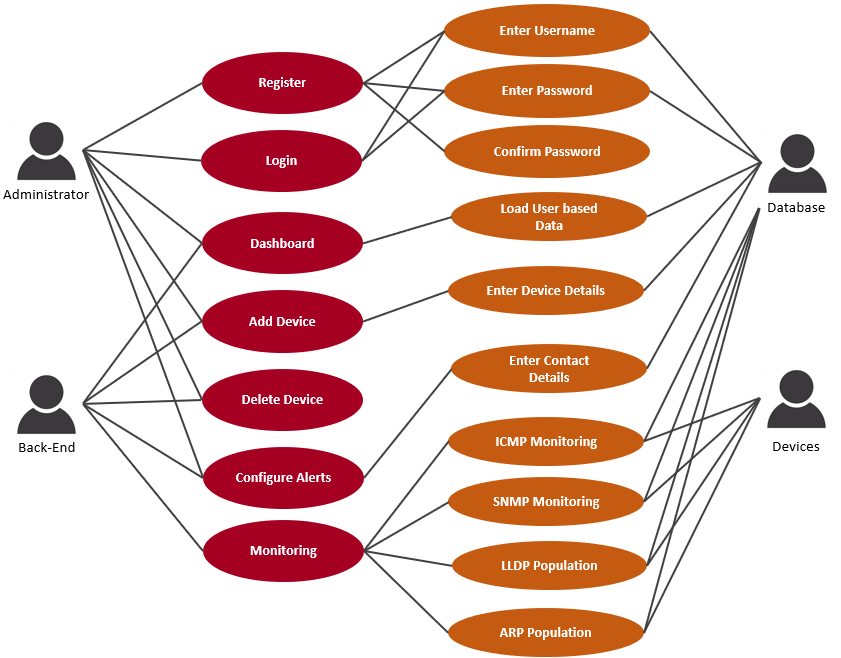
**4.2 Design:**

The various diagrams that illustrate the design of the network monitor are shown using use case diagrams, class diagrams and sequence diagrams in the following section:

**4.2.1 Use Case Diagram:**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams.

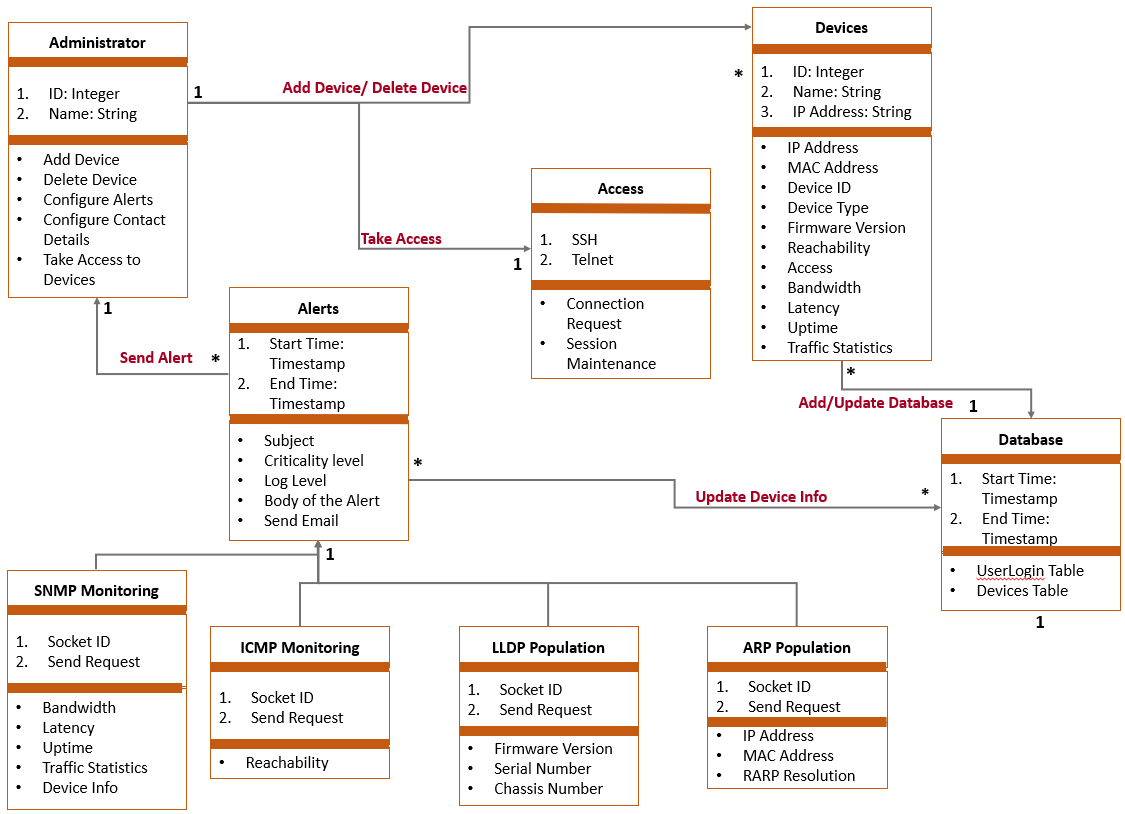
The Figure 4.2 shows the Use Case Diagram for Network Monitor that illustrates the relationship between the user and the system in different use cases. Various actors involved in the product are Administrator – user who controls the network monitor system, Back-end – involves a python script running the functions, Database – to store and relay the obtained data from devices and Devices that are monitored as per administrator’s instruction.

****

**Fig: 4.2 Use Case Diagram for Network Monitor**

**4.2.2 Class Diagram:**

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects. The Figure 4.3 shows classes and associations involved in Network Monitor.



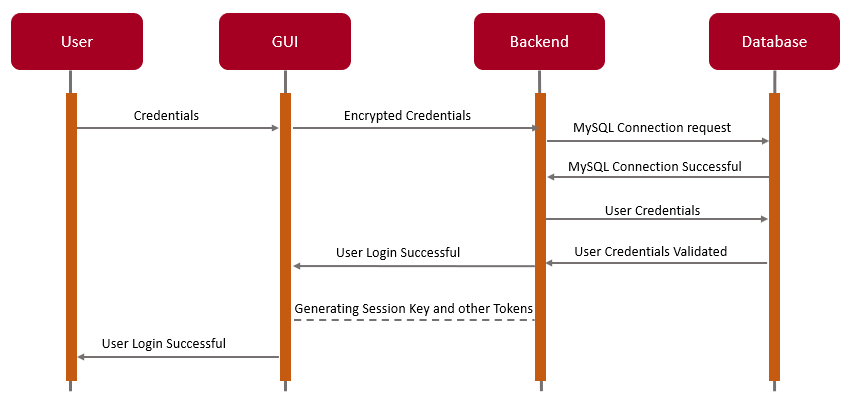
**Fig: 4.3 Class Diagram for Network Monitor**

The most important classes in this prohect are the Administrator, Devices, Alerts, Database, Access, SNMP Monitoring, ICMP Monitoring, LLDP Population, ARP Population.

1. The **Administrator** class contains various attributes such as ID, Name and so on which are needed to login to the Network Monitor System to perform various functionalities such as Adding a Device, Deleting a Device, Configuring custom Alerts, Configuring contact details such a email and so on and taking access to the device.
2. The **Devices** class containes several attributes such as ID, Name, IP Address and so on that are used to populate database with various parameters of each device such as MAC, Address, Device ID, Device Type, Firmware Version, Reachability, Access, Bandwidth, Latency, Uptime, Traffic Statistics and so on
3. The **Alerts** class contains two timestamps namely Start Time and End Time that identify every alert event. Information parameters that describe an alert include Subject, Criticality Level, Log Level, body of the alert and Sending an Email to configured mail address by the Administrator.
4. The **Access** class is used to denote enabling the administrator to take access to the devices using protocols such as SSH and Telnet provided the requisite access has been allowed and configured in device.
5. The **SNMP Monitoring** class contains attributes such Socket ID and Send Request that are used to obtain information related to the device including Bandwidth, Latency in reaching the device, Uptime of the device, Traffic Statistics in real time and information about device.
6. The **ICMP Monitoring** class contains attributes such Socket ID and Send Request that are used to determine the reachability of the device.
7. The **LLDP Population** class contains attributes such Socket ID and Send Request that are used to learn more about the device through the LLDP RX packets; firmware versions, serial number and chassis number are of interest in particular.
8. The **ARP Population** class contains attributes such Socket ID and Send Request that are mainly used to populate MAC Address and IP Address of the device into the Database. Occasionally this functionality can also be used to perform a RARP resoultion or even detect Gratiutious ARP packets as a future enhancement.

**4.2.3 Sequence Diagrams:**

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. Figure 4.4 explains how an administrator logs into the Network Monitor System. Various steps performed for this case are explained below:

****

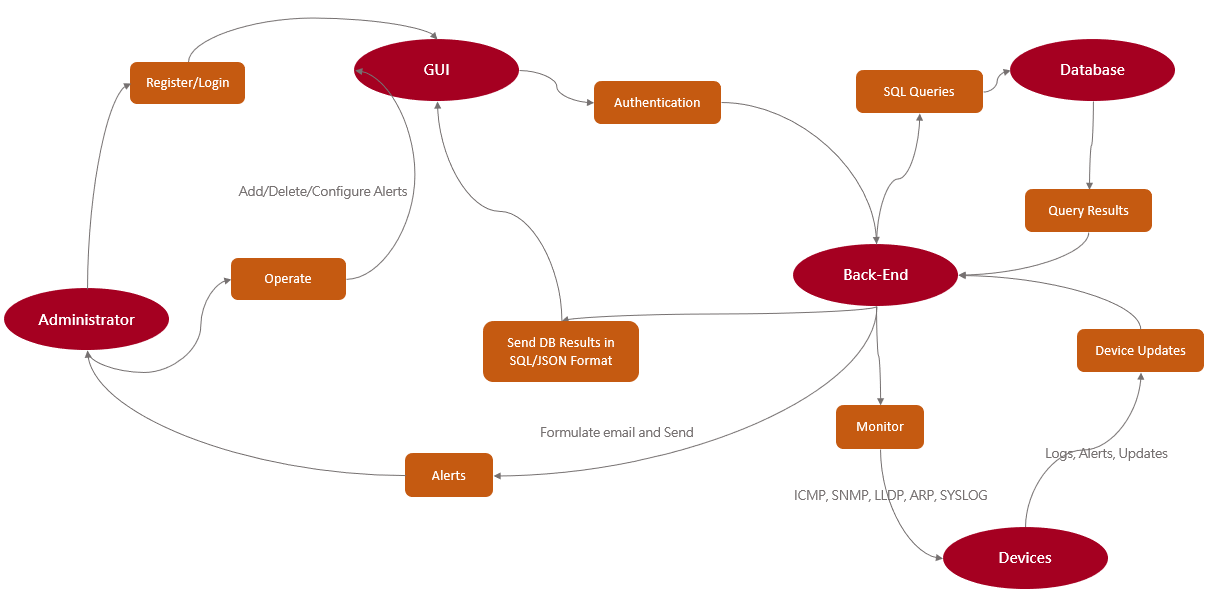
**Fig: 4.4 Sequence Diagram for Network Monitor**

Firstly, the user clicks on the **“Login”** button and enters the credentials namely username and password in the graphical user interface and clicks on **“Submit”** button. These credentials are then encrypted by the Django framework’s session manager and sent to the Backend python script. The Backend control makes connection request to the Database and sends the user credentials upon connection establishment. The Database, MySQL in this case, would verify the credentials and sends a “OK” message if valid as a query response. The Framework then generates session key and other tokens to manage the user session. Upon successful creation of the session, a message is displayed on the GUI that “User Login is Successful” and loads the customer specific dashboard.

**4.2.4 Data Flow Diagrams:**

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel.

** Fig: 4.5 Data Flow Diagram for Network Monitor**

The Figure 4.5 shows the various data items flowing for Network Monitor. The various data items involved in this design are: Authentication, addition/deletion of devices, alerts, database updates, query retrievals, device monitoring and device data updates.

Authentication involves registration and login modules of the network monitor. When the user enters credentials into the GUI and clicks on Login, the credentials are encrypted by backend processing and sent to the Database through SQL Query exchange. When the database validates the authentication request, the backend creates a new session along with their tokens. Then appropriate message is displayed on GUI. Adding/Deleting a device basically involves addition/removal of rows in Device Table of the database which are identified using the device id, name, and IP address as parameters.

An added device is monitored using different protocols, namely, ICMP, SNMP, LLDP, ARP and SYSLOG in the initial phase. Each protocol is used to obtain a particular data about various parameters attributing to the device. For example, ICMP provides us the reachability status and TTL values for required to reach the device. Similarly, LLDP is used to obtain Serial number, chassis number, firmware version and so on from the LLDP RX packets from the device. Administrators are facilitated with various options to configure alerts such as allowing only critical alerts to be sent at certain times. Based on this configuration and criticality of the events, various logs are accumulated and sent in email to the user. Backend processing is basically one constantly running python scripts that updates a database using pymysql framework, which acts as a buffer for GUI to load quickly, every time an update comes up from added devices.

**Chapter 5**

**APPLICATIONS**

* Networks serve as the backbone for any enterprise. Any network outage during working hours is huge loss for the organizations.
* As a result, they employ a separate team to look after their labs by constantly logging into several system interfaces and checking their statuses.
* It is a tedious task to login to each of these nodes, check if they are reachable and check their health status constantly.
* As the enterprise grows, the numbers increases exponentially.
* Generating network performance reports
* Deploying new technology and software upgrade successfully
* Monitoring the flow of traffic with netflow
* Track user network activity