Network Services Automation: An Efficient Service Provider IP Network Management System

By

Borona Silas Kimathi

079199

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Telecommunication Innovation at Strathmore University

School of Computing &Engineering Sciences

Strathmore University

Nairobi, Kenya

June 2025

Declaration and Approval

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student Name: Borona Silas Kimathi

Signature..... Date.....30/5/2025

Approval

This dissertation of Borona Silas Kimathi has been reviewed and approved by the following:

Dr. Kennedy Ronoh,

Senior Lecturer, School of Computing & Engineering Sciences,

Strathmore University

Dr. Julius Butime,

Dean, School of Computing & Engineering Sciences,

Strathmore University

Prof. Bernard Shibwabo,

Director of Graduate Studies,

Strathmore University

Abstract

In the rapidly evolving digital landscape, service providers face challenges in efficiently delivering data services due to manual processes i.e., service provisioning, cybersecurity compliance, and network inventory. To address this, the research focused on automating service provider IP network infrastructure processes by integrating technologies and methodologies to enhance efficiency.

The study begins by examining the complexities of current service provider infrastructure systems, highlighting the limitations and constraints of existing network manual processes and automation software. Manual processes lead to prolonged service configuration times, errors prone network configuration, poor network insights, increased operational costs, etc.

The developed solution constitutes an automated network system for service provider IP core network infrastructure. The system leverages Agile software methodology, and network automation using Python, and Django web framework. The core values and principles of Agile methodology, such as early and continuous delivery, working with smaller tasks that can be completed quickly, and welcoming changing requirements, were integrated into the system design.

The network management system developed in this research offers a comprehensive suite of tools for optimizing and automating service provider network infrastructure operations. Key features include the integration of Python and Django for robust web-based management system, leveraging libraries like Netmiko aimed at managing vendor-agnostic network environment. The system supports network cybersecurity compliance, reporting, inventory management, real-time network insights, network performance, intuitive topology dashboard and service provisioning. Additionally, it utilizes Celery for asynchronous task execution, enabling background processing of network compliance and device configuration updates. This combination of technologies enhances operational efficiency, reduces manual errors, and improves network reliability, contributing significantly to the scholarly discourse on network optimization and offering practical solutions for industry practitioners

Keywords: *Service Provider, Network Automation.*

Table of Contents

List of Figures

List of Tables

List of Abbreviations

TERM	MEANING
BGP	Border Gateway Protocol
CRUD	Database function Create Read Update and delete
CSS	Cascading Style Sheets
DFD	Data Flow Diagram
DRY	Do not Repeat Yourself
ERD	Entity Relationship Diagram
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IP	Internet Protocol
IP/MPLS	Internet Protocol Multiprotocol Label Switching
IPv4	Internet Protocol version 4
IPv6	Internet protocol Version 6
IS-IS	Intermediate System to Intermediate System protocol
ISP	Internet Service Provider
JS	JavaScript
KPI	Key Performance Indicators
LLDP	Link Layer Discovery Protocol
MNOs	Mobile Service Provides
MTV	Model template view
MVC	Model View Controller
ORM	Object-Relational Mapping
OSPF	Open Shortest Path First
SEO	Search Engine Optimization
SNMP	Simple Network Management Protocol
SRV6	Segment Routing IPv6
VLAN	Virtual Local Area Network
YAML	yet another markup language

Definition of Terms

TERM	MEANING
Django	Django is a free and open-source, Python-based web framework that follows the model–template–views architectural pattern(S. Chen et al., 2020).
Netconf	Network Configuration Protocol is a network management protocol that securely manages network devices such as routers, switches, and firewalls. It allows for the installation, manipulation, and deletion of configuration data on these devices (PyNetlabs, 2023).
Netmiko	Netmiko, built on top of Paramiko, is a Python library specifically designed to streamline the management of network devices via SSH, supporting a diverse array of vendors (Osei-Wusu et al., 2025).
Nornir	is an automation framework written in python to be used with python(Byers, 2022).
Paramiko	Pure-Python implementation of the SSHv2 protocol that offers both client and server capabilities (Paramiko, 2024).
Scrapli	Python library designed for establishing connections to various network devices, including routers, switches, firewalls, and more, through SSH or Telnet protocols (Montanari, 2020).

Acknowledgements

I express my profound gratitude to the Almighty for blessing me with good health, patience, and unwavering determination, which have been instrumental in the successful completion of this dissertation. I am deeply thankful to the Lord for granting me the resilience and vigor to navigate through the entire process of researching and writing this dissertation.

I extend my heartfelt thanks to my esteemed mentor, Dr. Kennedy Kibet Ronoh and Dr Vitalis Ozianyi whose invaluable guidance, support, insightful ideas, and unwavering motivation significantly contributed to the development and acceptance of this dissertation. My gratitude also extends to all the lecturers at MTI and my fellow classmates, whose encouragement and assistance proved to be invaluable throughout this academic journey.

Lastly, but certainly not least, I am profoundly thankful to my colleagues at Safaricom Limited and my family members. Their unwavering support, understanding, and encouragement have been a source of strength, enabling me to undertake this academic endeavor with determination and focus.

1: Introduction

1.1 Background

The increasing complexity and expansive nature of modern network infrastructures, predominantly within the domain of service providers, highlight the critical imperative for sophisticated network management solutions. As networks scale in size and variety, the demand for effective management becomes principal to guarantee optimal performance, cybersecurity, and resource utilization (Imarc, 2024).Traditional network management, characterized by manual network configurations, network performance monitoring, cybersecurity management, service provisioning and deployment, and reactive troubleshooting and diagnostics, is proving inadequate in accommodating the dynamic nature of contemporary networks.

While partial automation tools (such as Ansible, Chef, and Puppet) are already in use within some service provider networks, these solutions are often limited in scope, require significant manual oversight, and do not fully address the complexity and scale of modern Kenyan network environments and other similar African countries (Arinze, 2024). This study is concerned with developing and evaluating more advanced automation approaches that go beyond current practices, aiming to deliver end-to-end automation, improved integration, and enhanced security tailored to the unique needs and challenges of service providers in Kenya and similar African contexts.

The traditional, reactive approach to network management is no longer suited in an era where downtime and performance concerns can lead to significant losses. It is time to shift our outlook and adopt a proactive view toward network management. The limitations of established approaches are evident in their lack of ability to proactively address problems, adapt to evolving traffic patterns, and seamlessly integrate emerging technologies. The growing gap between the demands on service provider networks and the capabilities of traditional management strategies necessitates a paradigm shift toward automated and intelligent solutions. The inadequacies of orthodox network management methods are becoming increasingly pronounced, making the transition to more sophisticated frameworks imperative for enhancing operational efficiency and preserving the reliability and competitiveness of service provider network (Rakissaga et al., 2025).

In response to the limitations of traditional network management, there is an expanding interest in advanced solutions that leverage automation, machine learning, and artificial intelligence to augment network performance and reliability (Uchenna Joseph Umoga et al., 2024). The number of devices in networks and their diversity are consistently growing. The traditional methods employed for configuring network equipment are time-intensive, especially when factoring in the vendor-specific expertise required. Network automation through Python for instance, utilizes high-level business policies to guide network configuration and behavior, providing a more intuitive and business-centric approach to network management (Preeti Tupsakhare, 2019). This advanced approach empowers service provider networks to autonomously optimize their performance, predict and preempt issues before they arise, and dynamically adapt to changing network conditions. As service provider networks continue to evolve, embracing advanced network management strategies it becomes necessary to ensure scalability, efficiency, and the capability to converge the diverse demands of modern communication services.

The shift to advanced network management solutions is not merely theoretical; it is a practical necessity given the ever-evolving landscape of communication technologies and services. The advent of 5G networks, the proliferation of edge computing, and the accelerating complexity of multi-domain network architectures present unique challenges that mandate innovative and automated solutions. Research and development efforts in network management are thus critical not only to address existing challenges but also to get ahead and prepare for future requirements. Adoption of these advanced network management approaches is instrumental in establishing a foundation that can support the diverse and dynamic communication needs of modern societies (Bolettieri et al., 2022).

This dissertation aims to bridge the gap between theoretical understanding and practical implementation. It endeavors to offer a comprehensive understanding of how advanced automation techniques, seamlessly integrated into Service Provider IP network infrastructures, can usher in a unique era of data service delivery. Through careful analysis, real-world case studies, and innovative simulations, this research seeks to demonstrate the transformative impact of automated solutions on network configuration, cybersecurity management, performance, resource utilization, and overall network visibility (Sheth et al., 2023).

1.2 Problem Statement

Kenyan service providers increasingly struggle to manage the evolving challenges of modern IP networks using traditional, manual approaches. Manual configurations, inventory tracking, and reactive troubleshooting introduce significant bottlenecks, resulting in delayed issue resolution, prolonged downtimes, and frequent service disruptions. These challenges are further compounded by the lack of user-friendly interfaces, limited development agility, and inadequate cybersecurity auditing and hardening capabilities. As a result, the reliability and efficiency required for today's dynamic network environments are often compromised (John, 2021).

Although advanced network automation solutions are widely adopted in many countries, Kenyan and African service providers face unique obstacles such as infrastructure gaps, resource constraints, and regulatory complexities, which slow the adoption and limit the effectiveness of automation (Saivic, 2024) Existing automation tools including Ansible, Chef, Salt, Nornir, Scrapli, FastAPI, Puppet, Napalm, Netmiko, and Paramiko also present notable limitations. Many of these tools have steep learning curves, require extensive programming knowledge, or rely on complex architectures that can overwhelm users (Sharma, 2025). Most focus on automating isolated tasks rather than providing comprehensive, integrated solutions, necessitating ongoing human oversight and increasing the risk of errors and outages. Furthermore, some tools demand deep technical expertise, rely on central servers that create single points of failure, or operate within closed ecosystems that hinder integration and scalability (Muhammad & Munir, 2023).

The growing scale and complexity of service provider networks further overwhelm traditional and partially automated management methods. The sheer volume of devices, connections, and data traffic now exceeds what manual processes can efficiently handle, leading to suboptimal resource utilization and heightened cybersecurity risks (Huawei, 2024a). The reactive nature of legacy methods also makes it difficult to maintain performance and security standards amid rapid technological change. Slow adaptation to innovations like cloud computing and IoT further limits the ability of service providers to compete and innovate effectively (Kentik, 2025).

Addressing these shortcomings is essential for transitioning to a modern, efficient, and secure network management paradigm that meets the demands of contemporary IP networks in Kenya and similar contexts. This research seeks to bridge these gaps by proposing and evaluating

advanced, context-specific automation solutions tailored to the needs of African service providers.

1.3 General Objective

The overall objective of this research was to develop and implement a robust and intuitive network automation management system that enables service provider engineers to efficiently manage daily IP network operations. Key distinguishing features include user-friendly interfaces, enhanced operational efficiency, customized equipment management, tailored cybersecurity compliance views, detailed inventory management, and flexible reporting capabilities. Additionally, the system offers concise, real-time network insights immediately, addressing common limitations of existing solutions such as complexity and insufficient real-time visibility. By combining ease-of-use with tailored functionality, this system significantly reduces manual errors and enhances productivity in network management tasks.

1.4 Specific Objectives

i. To identify the challenges encountered with manual service provider network management.
ii. To review current solutions and frameworks for automating service provider network management.
iii. To design and develop an improved service provider network management system.
iv. To test automated service provider network management system.

1.5 Research Questions

i. What challenges are associated with manual network management?
ii. What solutions and frameworks currently exist for automating network processes?
iii. How can a network automation solution be designed and developed for effective network management?
iv. How was the automated service provider network management system solution be tested?

1.6 Scope

This focuses on exploring the application of advanced automation technologies, specifically network automation using Python and the Django web framework, within service provider network infrastructures (Huawei, 2024a). It provides an in-depth analysis of challenges faced by service providers in optimizing service efficiency, emphasizing aspects such as service

provisioning, configuration management, implementing cybersecurity controls, network virtualization, resource utilization, and network troubleshooting. Additionally, the study evaluates the effectiveness of the implemented automated processes within service provider network infrastructure and conducts a comprehensive assessment of user experience.

1.7 Justification

Challenges in service provider IP network infrastructure arise from traditional methods like manual configuration and network management, highlighting the critical role of network automation in modern IT operations. Leveraging software programs, scripts, and APIs streamlines tasks, enhancing security, reducing errors, and improving efficiency. This automation boosts service provider's operational effectiveness, scalability through rapid deployments, cost savings from reduced manual labor, enhanced agility in responding to changing business requirements and ensures quicker responses to security threats and streamlined network inventory management (Ramesh et al., 2023).

Recent studies have examined network automation solutions to tackle various challenges. For example, (Datta et al., 2023) highlighted Ansible's effectiveness in standardizing processes and automating configurations within ISP environments using YAML templates, noting gaps in comprehensive end-to-end network model coverage and integration of protocols like OSPF, ISIS, and BGP. Similarly, explored Nornir, Scrapli, and FastAPI integration for flexibility, scalability, and interoperability across network devices, while not addressing critical aspects such as network resource inventory, cybersecurity hardening, and configuration management.

These studies highlight the critical role of advanced automation technologies like Python and the Django web framework in optimizing service provider IP networks, enhancing configuration, network management, operational efficiency, and receptiveness to meet modern standards of network optimization.

1.8 Limitations

This study is subject to several limitations. The primary data was obtained from survey responses rather than direct observation of network processes, which may affect the depth and accuracy of operational insights. Furthermore, the research was conducted within a single Kenyan service provider, so the findings may not be directly generalizable to other firms or regions. The

automation design proposed here is tailored to the specific context of the studied organization and may need adaptation for broader application. Time and resource limitations also restricted the breadth of data collection and system testing. Future research should include observational data and involve multiple organizations to improve generalizability and robustness.

2: Literature Review

2.1 Introduction

In the changing world of network management automation is now seen as an element for the success and effectiveness of organizations (Sarrigiannis et al., 2022). With networks becoming varied, traditional methods of managing them are no longer sufficient causing challenges for many companies. The drawbacks of network management tools including costs, complicated setup procedures and demanding upkeep requirements have worsened the situation. This emphasizes the pressing need for automated solutions to address these issues.

The significant increase in network installations since the 1980s has resulted in a wide array of network technologies that require specialized knowledge for day-to-day operations. This growth has led to intricate and diverse network environments that are increasingly hard to handle. Modern network management involves planning, designing, optimizing performance modeling capacity and reporting, on service level agreements. To meet these demands effectively and efficiently a thorough automated approach is necessary (Wang et al., 2018).Network automation tools have become essential in simplifying operations reducing errors caused by humans and significantly improving the efficiency and dependability of network management processes.

In today's changing world managing complex network infrastructures poses significant challenges, for organizations. Embracing automation solutions offers an opportunity to revolutionize network operations. By utilizing these tools companies can tackle network management difficulties. Position themselves for future success and innovation in the digital realm. The shift towards automated network management is not an upgrade in technology; it is a move for organizations aiming to stay ahead in a swiftly evolving digital environment. Managing networks is pivotal in enterprise IT setups moving from error prone processes to sophisticated automation practices (Sarrigiannis et al., 2022).Given the complexities of networks relying on tools is essential to support network managers in their day-to-day tasks transforming them from reactive problem solvers into proactive planners. The International Organization, for Standardization (ISO) has outlined five areas of network management known as Fault, Configuration, Accounting, Performance and Security management (FCAPS). This framework underscores the importance of tools that effectively manage each aspect of network operations amidst the changing digital landscape. In this environment, effective network management requires optimization and adaptation to meet evolving user needs (Wang et al., 2018).

Organizations must have the capability to quickly set up oversee and adjust network operations to address business requirements and combat cybersecurity risks effectively (Wang et al., 2018). Using management utilities and automation network administrators can boost effectiveness and foster creativity guaranteeing that networks bring substantial benefits to both administrators and users. Figure 2.1 below demonstrates a basic network management architecture.

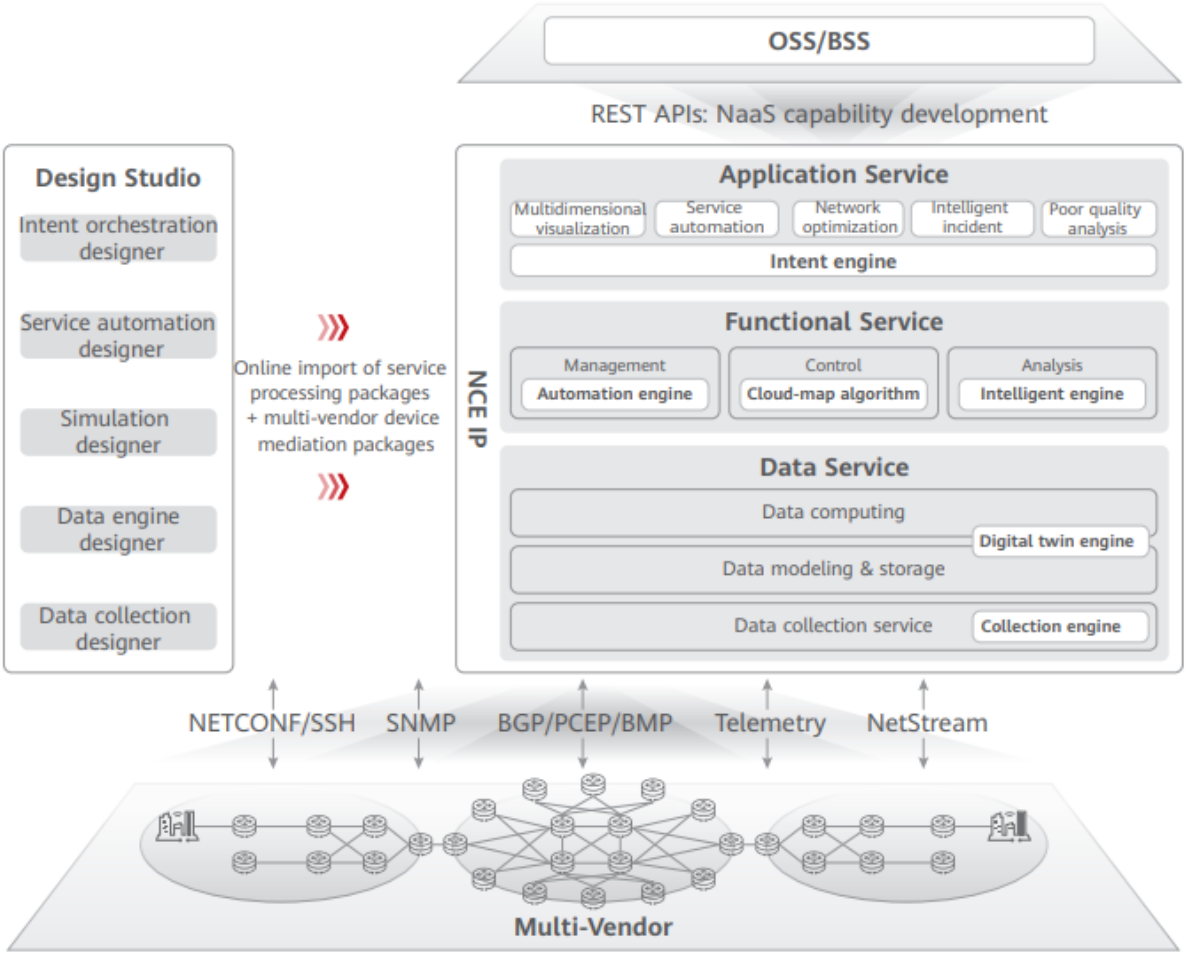


Figure 2.1: Network management architecture (Huawei, 2024)

2.2 Relevant Theories

The automation of IP network infrastructure is fundamentally supported by various theories and frameworks that highlight the necessity of dependable and effective network

systems. These theoretical foundations not only guide practitioners in implementing effective automation strategies but also represent substantial advancements in scholarly research (Gooley et al., 2023).A robust theoretical framework serves as a catalyst for innovation, enabling organizations to optimize network performance and adapt to the complexities of modern digital environments (Edelman et al., 2023). These frameworks are essential for fostering resilient and agile network management practices that can meet the evolving demands of today's technology landscape.

2.2.1 Network Automation Theory

The automation of IP network infrastructure is supported by various theories and frameworks highlighting the necessity of reliable and efficient network systems. These theoretical foundations not only guide practitioners in implementing effective automation strategies but also represent important advancements in scholarly research (Edelman et al., 2023). A robust theoretical framework catalyzes innovation, enabling organizations to optimize network performance and adapt to the complexities of modern digital environments (Gooley et al., 2023). These frameworks are essential for fostering resilient and agile network management routines that can meet the evolving demands of today's technology landscape.

The Network Automation Theory has emerged as a cornerstone for implementing and maintaining modern network infrastructures, offering a compelling framework for organizations to revolutionize their network management practices. This theory posits that the strategic implementation of automation tools and frameworks can lead to significantly more agile and resilient network infrastructures, a proposition that has been robustly validated through experiential data collected from network administrators (Muhammand & Munir, 2023). The theory's alignment with real-world practices underscores its practical value, providing a strong validation for firms to transition towards automated network systems, thereby addressing numerous challenges associated with modern network infrastructures (Bellamkonda, 2023).

2.2.2 Systems Theory

Systems theory can also describe network infrastructure as a complex system comprising interconnected components that function together to achieve specific objectives. Systems theory underscores the importance of understanding the interdependencies between various elements such as hardware, software, protocols, and human operators in network automation. Automation

can be seen as a mechanism to optimize the effectiveness and consistency of the network system by reorganizing processes, reducing manual involvement, and enabling adaptive responses to dynamic changes in network conditions. By applying Systems theory principles, network administrators can design and implement automated solutions that enhance service provider IP network overall performance and resilience (Muhammad & Munir, 2023).

2.2.3 Control Theory

Control Theory is another leading theory that looks at the role played by feedback in maintaining system stability and efficient functioning. For network automation purposes, this means monitoring network performance constantly and adjusting parameters to ensure smooth operation. Loops can be used to decrease issues, track better service quality, and adapt to changes in the environment. Applying Control theory principles enables providers to establish an automated system that catches and deals with early problems, thereby improving customer's dependability, scalability, and satisfaction on their networks. These theories are instrumental when it comes to planning, designing and implementing automated service provider IP networks helping professionals make the most of network automation (Zhang & Quan, 2022).

2.3 Benefits of Network Automation

Network automation can be depicted as reducing or eliminating human errors, enhancing network security, and increasing efficiency. A Jinja template engine can simplify the network configuration tasks while facilitating the standardization of the network infrastructure. Network administration activities are represented by frequent scalability challenges, which increase with the upsurge in the number of devices in each network (Santyadiputra et al., 2021). Therefore, scalability may not be feasible with traditional network management techniques. Standard techniques would require logging into the network devices individually, which would be inefficient.

2.3.1 Improved Efficiency

Consequently, the first benefit of network automation is a massive increase in operational effectiveness. Network administrators can therefore concentrate more on more strategic and complex activities by doing away with repetitive and time-wasting operations such as device configuration changes, software updates, and device provisioning. This alteration improves productivity and enables fast deployment of network services and applications. As a result, these

procedures are made seamless by automating them with tools. It takes less time to do routine work because these tools respond faster to network issues or any amendments in their design (Muhammad & Munir, 2023).

2.3.2 Reduced Human Errors

Many human mistakes in manual network management cause major failures and security faults. Network automation reduces these flaws with consistency in configuration and policy enforcement. These automated systems implement standardized configurations across the networks, lowering the chances of network performance and security being tuned through misconfigurations. Manual configuration in large-scale networks is not feasible. Accordingly, standardization is necessary, particularly since it reduces errors (Bringhenti et al., 2024).

2.3.3 Enhanced Network Security

Organizations are concerned about network security and its enhancement through automation. Through automation, security policies will be enforced evenly within the whole network, thus leading to the detection of threats that pose risks to networks as they happen, contrastingly in manual systems, which take longer to detect and respond. It is also a necessity that it is compliant with industry standards. For illustration, automated vulnerability detection and user activity tracking provide the network administrator with a means of monitoring access, implementing suitable access rights, and blocking unauthorized modifications. By automating these security tasks, organizations can significantly reduce the risk of a security breach and have a more secure network environment (Gulati et al., 2024).

2.3.4 Cost Savings

Bringing network automation will save costs by reducing expensive manual lead-ins and, therefore, make it less prone to errors. For example, automated systems need less human intervention to take care of typical tasks so that companies can dedicate their personnel to more strategic work. This can lower the costs of labor and increase resource optimization, making this a cost-effective solution. In addition, automation is further useful to optimize the network performance by saving costly operations incurred when networks events occur (Edelman et al., 2023).

2.3.5 Scalability and Flexibility

Another critical advantage of network automation is scalability. With automated systems, scaling of network infrastructure either up or down becomes considerably more straightforward to accommodate growth or adapt to changing demands. Flexibility is very important for an organization that needs to respond quickly to a change in the market or an evolution in technology. Network automation will be able to ensure that new devices and services are integrated into the system without any hitch, thus ensuring seamless growth and evolution of network environments without too much manual intervention (Saeik et al., 2021).

2.3.6 Improved Network Performance and Monitoring

The process employs automated monitoring and analysis of traffic on the network to help discover bottlenecks or other performance issues efficiently. Automation tools give you a live view of your network state, performance and security by activating alerts and responses following predefined thresholds or patterns. Proactive network management not only addresses possible issues before they affect the operations of networks but also results in a more dependable and high-performing system of networking environment (Choudhury, 2024).

2.4 Challenges and implications of manual network management systems

The transition from manual to automated network management systems is critical for service providers grappling with the limitations of traditional approaches. Manual network management systems are plagued by inefficiencies such as lack of real-time visibility into asset status, manual service provisioning and network performance, leading to prolonged fault resolution times and reactive troubleshooting. Reliance on spreadsheets and siloed databases exacerbates human error rates, particularly during scaling efforts, resulting in misconfigurations, service disruptions, and compliance risks. Legacy methods also struggle with scalability challenges, as evidenced by the inability to manage growing 5G and IOT complexity, which increases operational expenses (OPEX) due to redundant workflows and delayed deployments. Furthermore, the absence of integration with existing systems creates operational silos, hindering correlation between infrastructure data and customer service requirements. These shortcomings underscore the urgent need for automation, as manual processes fail to meet modern demands for agility, precision, and cost-efficiency in IP network management (Ramesh et al., 2023).

2.5 Models and Frameworks

Improvements in technology today have exceptionally enhanced the service provider networking environment by increasing the number of devices and tasks per system. Automation tools play a vital role in the extension of the operation of the network. This provides a framework to assist developers in constructing effective and scalable network architectures. Unique features represented by automation tools assist in developing robust network systems that improve efficiency.

Network automation tools are about having a reasonable edge; in other words, automation modernizes network management and enhance cybersecurity. Ansible, Puppet, and Chef have strong configuration management and software deployment capabilities that help IT teams meticulously manage complex infrastructures. Due to increasing in network softness and cloudification, automation tools are required to make new technologies like intent-based networking implementable. Therefore, automation ensures robust cybersecurity, optimal performance, and the ability to swiftly adapt to changing business requirements (Ojha, 2024).

2.5.1 Automation Orchestration and Provisioning

Automation and orchestration enhance the speed and reliability of the end-user of resources and services. It reduces the need for intervention and subsequent errors to improve efficiency. Ansible and Terraform are tools that enable automatic provision of network resources. For instance, terraform provides Amazon Web Services (AWS) with automation processes for setting up instances and network settings to ensure scalability with efficient resource management. Consul Terraform Sync automates network infrastructure to ensure safety and security since changes are handled automatically with events triggered by service growth, address, or port number change, or even tag changes. This results in deliveries and reduced configuration mistakes (Gulati et al., 2024).

2.5.2 Intent-Based Networking (IBN) Systems

Network automation is one of the sure strategies for a firm to enhance network operations while sustaining the competitive advantage in the fast-paced technology context that has seized the world today. It makes a company operationally very proficient, further reducing human errors in general, by using advanced technologies and frameworks such as AI-driven chatbots and intent-based networking systems to obtain improved customer experiences (Njah et al., 2023).

For example, Cisco's DNA Center constitutes one of the network automation systems relative to business intent, easing the delivery of networks for enterprise services. Moving to such automated and intelligent management of a network not only improves scalability and cybersecurity issues but also leads an organization's response to be agile enough to keep up with rapidly changing business needs, which has been driving innovation and growth in a digital era (Clemm et al., 2022).

2.5.3 Machine Learning and Predictive Analytics

Machine learning and predictive analytics frameworks, such as TensorFlow and Apache Spark, allow service providers to estimate network demands in advance and, therefore, proactively create service optimizations. For instance, Google uses TensorFlow to build predictive models in which it analyzes the pattern of user behaviors to optimize content delivery and efficiently provide data services to its worldwide users. With such advanced capabilities, an organization can be able to look ahead and adjust operations to address the expected need, thereby raising service reliability significantly and achieving greater customer satisfaction (Bringhenti et al., 2024).

2.5.4 Self-Healing Networks

These self-healing network frameworks detect and remediate issues autonomously to ensure minimal downtime and optimal performance. Since Software for Open Networking in the Cloud (SONiC) works by self-healing networks, Microsoft Azure's networking infrastructure is resilient and self-healing, enhancing service delivery to its base. This innovative approach reduces downtime incidents and dramatically reduce the time to diagnose and fix network problems. Thus, one can realize network efficiency and reliability, translating seamlessly into perfect service delivery and great user experiences. The adoption of self-healing networks, as exhibited in Microsoft's recent implementation of SONiC, is the key step of network automation to revolutionize modern IT infrastructure management (Thota, 2025).

2.5.5 AI-Driven Customer Experience Management

AI-driven frameworks like chatbots and sentiment analysis tools have modernized customer interactions and significantly enriched service satisfaction across industries. For instance, T-Mobile's implementation of AI-powered chatbots has spectacularly improved customer query handling efficiency. The adoption of cutting-edge automated frameworks by

service providers, including automation orchestration, intent-based networking, predictive analytics, self-healing networks, and AI-driven customer experience management, has reshaped the landscape of IP core network infrastructures. These advanced approaches enable service providers to streamline operations and proactively address probable problems before they impact customers, eventually driving innovation, growth, and competitive advantage in today's rapidly evolving technological landscape (Singh, 2023).

2.6 Design and applications of Network automation tools.

Network automation tools have reformed how organizations manage, secure, and inventory their network infrastructure. This section explores the design principles and practical applications of several prominent network automation tools: Ansible, Chef, Salt, Napalm, Netmiko, Nornir, Scrapli and FastAPI.

2.6.1 Ansible for Network Automation

Developed by Red Hat, Ansible is an open-source automation tool that has become one of the most popular options in network automation due to its agentless architecture and user-friendly approach. It was created with flexibility and ease of use to offer a network automation platform. Ansible is agentless, so no software is needed on the managed devices; it uses SSH or API connections instead. This makes it easier to implement and reduces cybersecurity risks associated with such software (Bellamkonda, 2023).

It provides a human-readable configuration format through YAML-based playbooks that define automation tasks, which are easy to maintain, lowering the learning curve for network administrators. Ansible has a modular structure with roles and collections that improve the reusability and organization of its code. In this fashion, it improves efficiency in handling complex network environments. Moreover, extending any custom modules and plugins in the framework will enable an organization to scale Ansible for their requirements, hence flexibility towards different network infrastructures and needs. All these design principles make Ansible quite effective and an effective tool for simplifying the automation processes related to the network and enhancing operational efficiency. Ansible can automate cybersecurity policies across a network, pushing out the same firewall rules to thousands of devices with very little human intervention (Mahant & Kalapparambath, 2024).

Ansible, while a widely used automation tool, has several disadvantages that users should consider. It features a limited user interface that primarily relies on command-line operations,

which can lead to synchronization issues between the GUI and CLI. Additionally, Ansible lacks a notion of state, meaning it does not track dependencies and executes tasks sequentially without maintaining a detailed catalog, which can hinder complex automation processes. Furthermore, its support for Windows is limited, requiring a Linux control machine for management, which may complicate operations in mixed environments Generally, while Ansible is effective for many tasks, these limitations can impact its usability in certain scenarios (Bellamkonda, 2023).

2.6.1.1 Network Management Example:

Figure 2.2 demonstrates ansible playbook snippet exhibit displaying how to configure network time protocol servers on Cisco and Juniper devices.

```
- name: Configure NTP servers
  hosts: all
  tasks:
    - name: Configure NTP on Cisco IOS
      cisco.ios.ios_ntp:
        server: 10.0.0.1
      when: ansible_network_os == 'ios'

    - name: Configure NTP on Juniper
      junipernetworks.junos.junos_ntp:
        server: 10.0.0.1
      when: ansible_network_os == 'junos'
```

Figure 2.2 Playbook for multi-vendor integration (Killeen, 2025)

2.6.1.2 Network Cybersecurity Management:

Figure 2.3 illustrates how ansible enforces cybersecurity policies across the network. For example, it can automate the process of updating access control lists (ACLs) on network devices.

```yaml
- name: Update ACLs
  hosts: firewalls
  tasks:
    - name: Apply new ACL rules
      cisco.ios.ios_acls:
        acls:
          - name: INBOUND_ACL
            acl_type: extended
            aces:
              - grant: permit
                protocol: tcp
                source: 10.0.0.0/24
                destination: any
                dest_port: 80
```

Figure 2.3 Playbook applies a new inbound ACL (Killeen, 2025)

2.6.1.3 Network Inventory Management:

Figure 2.4 demonstrates ansible script to manage network inventory. Here's an illustration demonstrating the process of gathering and storing device information.

```yaml
- name: Collect device information
  hosts: all
  tasks:
    - name: Gather facts
      ansible.builtin.setup:

    - name: Store device info
      ansible.builtin.copy:
        content: "{{ ansible_facts | to_nice_yaml }}"
        dest: "/tmp/inventory/{{ inventory_hostname }}.yaml"
```

Figure 2.4 Playbook collecting detailed information in YAML (Killeen, 2025)

2.6.2 Chef for Network Automation

Chef, known primarily for managing server configurations, has broadened its scope to include network automation. Chef's design principles are carefully designed to create an adaptable automation framework for network management. While it offers agentless options for network devices, its agent-based architecture ensures control and monitoring capabilities. By using a Ruby-based Domain Specific Language (DSL) to write "recipes, " Chef enables easily

understandable configuration management appealing to both developers and system administrators (Katariya et al., 2025).

Chef's steep learning curve and complex initial setup can be challenging, particularly for those new to the platform. Additionally, Chef lacks push configuration, which many consider essential for managing operations at scale, as it primarily follows a pull-based process that adheres to predefined schedules. These drawbacks should be weighed against an organization's specific requirements when selecting a network automation tool (Katariya et al., 2025).

2.6.2.1 Network management:

Centralized management through a Chef server facilitates streamlined operations and consistent policy enforcement across the network infrastructure. Additionally, Chef's extensibility through custom resources and cookbooks allows organizations to tailor the automation framework to their needs, ensuring adaptability and scalability in diverse network environments. These design principles collectively enhance operational efficiency, reduce errors, and optimize service delivery, making Chef an imperative tool for modern network management.

Managing everything in one place using a Chef server helps keep things running smoothly and ensures that rules are followed consistently across the network setup. Likewise, by customizing resources and cookbooks, organizations can fine-tune the automation system to fit their requirements, assuring flexibility and growth in network setups. These guiding principles work together to boost how smoothly things run, minimize mistakes, and improve service delivery, making Chef an essential tool for handling networks (Katariya et al., 2025).

Figure 2.5 illustrates how chef can configure virtual local area network instance in a network device.

```
cisco_vlan '100' do
  vlan_name 'Web_VLAN'
  shutdown false
  action :create
end
```

Figure 2.5 Web_VLAN creation (Juniper Networks, 2022)

2.6.2.2 Network Cybersecurity Management:

In network cybersecurity management, Chef stands out for its ability to automate compliance checks and remediation processes, effectively lowering the chances of configuration inconsistencies and cybersecurity weaknesses. Chef can automate the enforcement of cybersecurity policies. For instance, figure 2.6 specific recipe guarantees that authorized SSH ciphers are utilized.

```
template '/etc/ssh/sshd_config' do
  source 'sshd_config.erb'
  variables(
    ciphers: ['aes256-ctr', 'aes192-ctr', 'aes128-ctr']
  )
  notifies :restart, 'service[sshd]'
end
```

Figure 2.6 Recipe updates the SSH configuration (Reintech, 2024)

2.6.2.3 Network Inventory Management:

Chef can be used to maintain an up-to-date inventory of network devices. The following recipe collects and stores device information. Figure 2.7 illustrates a configuration recipe that gathers essential device information and stores it in a JSON file, supporting effective inventory tracking and management.

```
node.default['network_inventory'] = {
  'hostname' => node['hostname'],
  'ip_address' => node['ipaddress'],
  'model' => node['dmi']['system']['product_name'],
  'serial' => node['dmi']['system']['serial_number']
}

file '/var/chef/inventory.json' do
  content JSON.pretty_generate(node['network_inventory'])
end
```

Figure 2.7 This recipe collects key device information (Gargano., 2020)

2.6.3 Salt for Network Automation

Salt, developed by SaltStack (now part of VMware), offers a flexible approach to network automation with its event-driven architecture.

Salt's innovative design principles offer a versatile and powerful approach to network automation. Its flexible architecture, supporting both agent-based (Salt minion) and agentless (Salt SSH) options, allows for seamless integration across diverse network environments. The use of YAML-based state files for configuration management ensures clarity and ease of use, reducing the learning curve for network administrators. Salt's event-driven architecture enables real-time responses to network changes, significantly enhancing operational agility and responsiveness. Furthermore, its extensibility through custom modules and runners allows organizations to tailor Salt to their specific needs, ensuring adaptability to unique network requirements. These design principles collectively make Salt a necessary tool for modern network automation, offering flexibility, efficiency, and scalability in managing complex network infrastructures.

Salt, created by Salt Stack, is part of VMWare and provides an approach to automating networks through its event-driven design. The innovative principles behind Salt offer an adaptable method for network automation. Its architecture, which supports agent-based (Salt minion) and agentless (Salt SSH) options, allows for integration in network settings. By using YAML-based state files for configuration management, Salt ensures clarity and user-friendliness, making it easier for network administrators to navigate. With its event-driven structure, Salt can respond quickly to network changes, enhancing agility and responsiveness. Moreover, the ability to customize Salt through custom modules and runners enables organizations to tailor it to their needs ensuring adaptability to network requirements. These design principles collectively establish Salt as a tool, for network automation that offers flexibility, efficiency and scalability in managing intricate network infrastructures.

SaltStack's inherent complexities and basic GUI can be limitations for some users, requiring a steep learning curve and Python knowledge. Additionally, SaltStack's execution time is sometimes longer compared to alternatives like Ansible. These drawbacks should be considered alongside an organization's specific requirements when choosing a network automation tool (Özdoğan et al., 2023).

2.6.3.1 Network Management Example:

Salt manages network device configurations across multiple vendors. Here's an example of a Salt State file for configuring NTP servers. Figure 2.8 depicts a configuration state that ensures the specified NTP servers are properly set on the network device, in accordance with recommended practices

```
configure_ntp:
  netconfig.managed:
    - template_name: salt://templates/ntp.jinja
    - servers:
        - 10.0.0.1
        - 10.0.0.2
```

Figure 2.8: NTP Servers configuration(Salt, 2024c)

2.6.3.2 Network Cybersecurity Management:

Salt can automate cybersecurity policy enforcement. For example, the following state file configures firewall rules. Figure 2.9 illustrates a configuration state that permits web traffic to a designated server through the firewall, ensuring appropriate access control.

```
configure_firewall:
  netacl.managed:
    - name: accept_web_traffic
    - family: ipv4
    - source: 0.0.0.0/0
    - destination: 10.0.0.1/32
    - protocol: tcp
    - dport: 80
    - action: accept
```

Figure 2.9: Firewall rules (Salt, 2024b)

2.6.3.3 Network Inventory Management:

Salt can maintain an up-to-date inventory of network devices. Figure 2.10 illustrates a configuration state that collects device facts and stores them in a YAML file, thereby facilitating effective inventory management.

```
collect_device_info:
  module.run:
    - name: net.facts

store_device_info:
  file.managed:
    - name: /var/salt/inventory/{{ grains['id'] }}.yaml
    - contents: {{ salt['yaml.encode'](salt['net.facts']()) | yaml_encode }}
```

Figure 2.10:A YAML file, facilitating inventory management (Salt, 2024a)

2.6.4 Nornir

Nornir was the newest, with a Python-native approach to network automation. It is especially good at highly complex network tasks where very granular control and custom logic are needed. Its application in network management scenarios, like automated network testing and validation, confirms its power in ensuring a network's reliability and performance.

Nornir's dependence on Python programming may pose a challenge for users who do not have programming skills and favor more intuitive interfaces. Moreover, Nornir is not naturally idempotent, necessitating extra effort to establish a read, compare, and write approach to achieve idempotency (Muhammad & Munir, 2023).

2.6.5 Scrapli

Scrapli is a Python tool that streamlines the process of connecting to and communicating with network devices. It offers a user-reliable interface for setting up devices running commands and fetching results. By providing an approach to device interactions Scrapli helps maintain the efficiency of operations. Widely recognized as a Python library for automating network tasks Scrapli ensures an intuitive understanding of interacting with devices, on various platforms.

One significant drawback is its reliance on Python, which may pose a barrier for users who lack programming skills, making it less accessible for network engineers accustomed to graphical interfaces. Additionally, Scapli does not provide built-in error handling or idempotency, which can complicate the automation process and lead to unintended consequences if not managed carefully. Furthermore, as a relatively new tool, it may lack extensive community support and documentation compared to more established automation frameworks, which can hinder troubleshooting and effective implementation (Gondhalekar et al., 2024; Montanari,

2020).Figure 2.11 shows how Scrapli cab used to interact with network device to retrieve configuration.

```python
from scrapli import Scrapli

device = {
    "host": "192.168.1.1",
    "auth_username": "admin",
    "auth_password": "password",
    "auth_strict_key": False,
    "platform": "cisco_iosxe"
}

with Scrapli(**device) as conn:
    response = conn.send_command("show version")
    print(response.result)
```

Figure 2.11: Retrieve device information (Montanari, 2020)

2.6.6 FastAPI

It has completely transformed the way web-based interfaces are developed for managing networks. With its top-notch performance user interface and built-in support for programming. FastAPI is perfect for building cutting-edge interactive web applications that cater to the needs of network administrators. Its API documentation generation and robust data validation capabilities simplify the development process, enabling prototyping and deploying network management interfaces. According to combining web frameworks like FastAPI with network automation tools can improve network management systems' efficiency and user experience. Leveraging FastAPI's power, organizations can craft tailored network management solutions that seamlessly integrate with their automation workflows, leading to more effective network operations.

One significant drawback is its reliance on Python type hints, which may pose a challenge for developers who are not familiar with Python or its typing system, potentially leading to a steeper learning curve. Additionally, FastAPI's asynchronous capabilities can introduce complexity in managing concurrent requests, which may result in difficulties for those not accustomed to asynchronous programming patterns. Moreover, as a relatively newer framework,

FastAPI may lack the extensive community support and resources available for more established frameworks, which can hinder troubleshooting and development efforts (Chen, 2023). Figure 2.12 demonstrates FastAPI capability to interact with network devices.

```python
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Device(BaseModel):
    name: str
    ip: str
    status: str

@app.get("/devices/{device_id}")
async def get_device(device_id: int):
    # In a real scenario, this would query a database or network device
    return {"device_id": device_id, "name": "Router1", "ip": "10.0.0.1", "stat
```

Figure 2.12: API endpoints creation (FastAPI, 2024)

2.6.7 Puppet

Puppet has become a tool for automating tasks in the open-source community. It allows users to centrally manage servers, networks, and storage systems using a language approach. This method lets administrators define how they want their systems to be set up, making it easier to read and maintain configuration code. The use of Infrastructure as Code (IaC) principles is well aligned with Puppets approach to helping organizations efficiently and consistently manage their infrastructure. Puppets architecture involves an agent server setup that gives control over managed nodes enabling configuration management and policy enforcement across different environments (Drosos et al., 2024).

The tool also benefits from a community that contributes to various modules and resources, making complex automation tasks more straightforward to implement. Despite its advantages, implementing Puppet may pose challenges for some users. Creating configurations might require coding skills, which could be daunting for new teams to use the tool. Additionally, Puppet agents can use up a lot of system resources during operations possibly impacting the performance of managed nodes. Despite the difficulties involved, Puppets' strong features shine

through in handling infrastructures and automating deployment duties. Its aptitude for maintaining uniformity across setups and linking with DevOps tools proves valuable for companies aiming to simplify their IT procedures and boost infrastructure dependability (Farayola et al., 2023).

One major drawback is its reliance on a centralized architecture, which creates a single point of failure; if the Puppet master server becomes unavailable, it can disrupt the management of the entire infrastructure. Puppet's agent-based model can be resource-intensive, requiring substantial central processing unit (CPU) and memory on managed nodes, which may not be ideal for smaller setups. The complexity of Puppet's Domain Specific Language (DSL) can also pose challenges for users unfamiliar with programming, making it less accessible for those without a technical background. Furthermore, Puppet's pull-based model can lead to delays in applying configuration changes, limiting its effectiveness for real-time automation needs (Özdoğan et al., 2023). Figure 2.13 presents a Puppet manifest that exemplifies the configuration of a network switch, including the setup of an Ethernet interface, the creation of a VLAN, and the configuration of a corresponding Layer 3 interface.

```puppet
# Network configuration using Puppet (Puppet, 2023)
node 'switch01.example.com' {
  network_device::interface { 'Ethernet1':
    ensure    => 'present',
    ipaddress => '192.168.1.1',
    netmask   => '255.255.255.0',
    speed     => '1g',
    duplex    => 'full',
  }

  network_device::vlan { '100':
    ensure    => 'present',
    vlan_name => 'Data_VLAN',
  }

  network_device::l3_interface { 'Vlan100':
    ensure    => 'present',
    ipaddress => '10.0.0.1',
    netmask   => '255.255.255.0',
  }
}
```

Figure 2.13: Switch configuration manifest (Puppet, 2024)

2.6.8 Napalm

NAPALM, which stands for Network Automation and Programmability Abstraction Layer with Multivendor support, has emerged as a vital tool in network automation by offering an API that facilitates the management of devices from various vendors. It simplifies automation processes by standardizing network configurations. Allows for the management of diverse infrastructure. Its features for configuration management, such as retrieving, modifying, and validating configurations along with the ability to compare states, contribute significantly to ensuring network stability and consistency. Moreover, its seamless integration with Python-based automation frameworks enhances its usefulness in network environments, enabling engineers to make the most of its capabilities within automation systems (Gondhalekar et al., 2024).

NAPALM offers benefits for network automation tasks; it also poses challenges that network engineers need to be aware of. The library's reliance on protocols like NETCONF and SSH may restrict its use in environments that utilize older systems. Besides, mastering Python programming skills and understanding NAPALM functionalities might require time and resources for teams transitioning to this approach toward automation.

Nevertheless, despite these downsides, the extensive multivendor support and efficiency in simplifying network management tasks make NAPALM a dependable and effective tool for automating networks in the varied network environments of today. Figure 2.14 illustrates the process of network border gateway protocol configuration using the NAPALM library (Stancu et al., 2019).

```python
# NAPALM Script (NAPALM, n.d.)
from napalm import get_network_driver

driver = get_network_driver('ios')
device = driver('192.168.1.1', 'username', 'password')
device.open()

bgp_config = {
    'neighbors': {
        '10.0.0.2': {
            'remote_as': 65000,
            'description': 'BGP neighbor'
        }
    }
}

device.load_merge_candidate(config=bgp_config)
device.commit_config()
device.close()
```

Figure 2.14: BGP neighbor configuration (Napalm, 2024)

2.6.9 Netmiko

Netmiko simplifies network device interaction by providing a unified API for various network vendors. This tool is handy in automating routine tasks like configuration backups and software updates across heterogeneous network environments. Its simplicity and broad vendor support make it an excellent choice for organizations beginning their network automation journey. Netmiko simplifies interaction with network devices. Here's an example of connecting to a device and running a command (Osei-Wusu et al., 2025).

One significant drawback is that it primarily focuses on SSH-based interactions, which may limit its functionality with devices that do not support SSH or require different protocols. Netmiko lacks a built-in mechanism for error handling and idempotency, which can lead to issues when executing commands that may not produce consistent results across multiple runs. Netmiko's documentation can be sparse or unclear, making it challenging for new users to fully leverage its capabilities(Gondhalekar et al., 2024). Figure 2.15 illustrates the process of establishing a connection to a network device and retrieving information using the Netmiko library.

```
from netmiko import ConnectHandler

device = {
    "device_type": "cisco_ios",
    "host": "10.10.10.10",
    "username": "admin",
    "password": "password"
}

with ConnectHandler(**device) as net_connect:
    output = net_connect.send_command("show ip int brief")
    print(output)
```

Figure 2.15: Interface retrieval manifest (Vasudevan, 2025)

2.7 Gaps in current Network Automation Tools

The landscape of network automation tools, including Ansible, Chef, Salt, Nornir, Scrapli, FastAPI, Puppet, Napalm, Netmiko, and Paramiko, reveals several limitations that can hinder their effectiveness. Ansible, while praised for its simplicity, relies on YAML for playbooks, which can be challenging for users without programming experience and may lead to syntax errors. Similarly, Chef's Ruby-based Domain specific language (DSL) presents a steep learning curve, making it less accessible for network engineers who lack coding skills (Katariya et al., 2025). SaltStack's complex master-minion architecture can overwhelm new users, while Nornir's lack of a built-in user interface may deter those who prefer graphical interactions. Additionally, Scrapli's focus on SSH and Telnet connections limits its applicability in diverse environments, and FastAPI, though excellent for API development, is not primarily designed for network automation, which can complicate integration with existing tools (Klimai, 2022).

Moreover, many of these tools tend to focus on automating individual tasks rather than providing comprehensive solutions that reduce the need for human intervention. For instance, while Puppet and Chef automate specific workflows, they still require human oversight for script creation and error monitoring, which can lead to increased risks of errors and outages. Paramiko, although powerful for SSH connections, demands a deep understanding of SSH and command-line interfaces, creating a steep learning curve and requiring users to manage low-

level details themselves (Wang et al., 2018).Furthermore, the reliance on a central server in both Chef and Puppet introduces potential single points of failure, impacting the availability of configurations during outages.

Lastly, the flexibility and adaptability of these tools are often insufficient to keep pace with rapidly evolving technologies and methodologies. Most tools also operate within closed ecosystems that restrict integration with third-party solutions, limiting their functionality and scalability. For example, while Napalm simplifies network automation across different vendors, its dependency on specific device APIs can lead to compatibility (Gondhalekar et al., 2024). Additionally, the complexity of these tools can create a false sense of cybersecurity among network engineers, who may assume that automation will eliminate all potential issues without ongoing human oversight. Many tools utilize a command line interface, which can present a steep learning curve and is often not very user-friendly. Addressing these gaps is crucial for organizations aiming to enhance their network automation efforts and ensure they remain competitive in an increasingly digital landscape.

2.8 Related Case Studies on Developed Network Automation Systems

Exploring various similar research or case studies involving the development automation of network systems enables the researcher to understand the scope of the current project to avoid redundancy and apparent errors. Case studies demonstrating effective and efficient systems are crucial in inspiring the researcher to develop the most desirable project output. Such studies provide practical illustrations of how the researcher can implement the proposed system.

2.8.1 Case Study 1

Systems from the individual cases were built using stacks, templates, and architectures (Muhammad & Munir, 2023). developed an open-source network automation system leveraging Nornir, Scrapli, and FastAPI, which they explained as a cutting-edge solution for automating network device configuration, management, operation, and troubleshooting. The project aimed to automate as much of the network device lifecycle as possible, using Nornir for task implementation and inventory management, Scrapli for device connectivity, and FastAPI for creating a user-friendly web interface. To collect the necessary information for the project, the researchers employed both qualitative and quantitative approaches, including comprehensive literature reviews and empirical data collection through interviews and surveys with network administrators.

The project was designed to demonstrate the practical applications of automation tools in network administration and highlight their value in optimizing network operations. They analyzed various automation tools and their efficacy in managing network devices. These are aspects such as configuration, operation, and maintenance. Despite its strengths, such as providing a troubleshooting module and using open-source APIs, the project also had limitations, including the lack of a user interface, limited customization, and insufficient cybersecurity hardening modules. These insights underscore the importance of picking the right tools and frameworks to meet specific network automation needs and the potential challenges that may arise in their implementation (Muhammad & Munir, 2023).

2.8.2 Case Study 2

Network automation has become an essential strategy for organizations seeking to optimize their network operations, enhance efficiency, and maintain a competitive edge in today's rapidly evolving technological landscape. The automated network system implemented based on the Network Automation Theory, as described by (Santyadiputra et al., 2021) establishes the capacity of such systems to streamline network management and configuration tasks. This demonstrates how automation can substantially decrease the time and resources needed for regular network tasks while lowering the potential for human mistakes.

The system's design, which integrates a single administrator user type with comprehensive CRUD capabilities and device categorization by vendor, aligns well with best practices in network automation. As noted by (Edelman et al., 2023) such an approach allows for consolidated control and management of network devices. Thus, this is crucial for maintaining consistency and cybersecurity across complex network environments. The use of Python, along with libraries like Paramiko and Netmiko for SSH connections and configuration, reflects a growing trend in network automation towards leveraging powerful, flexible programming languages and open-source tools.

Although the system demonstrates several strengths, including database connectivity and a user-friendly Django framework interface, it also has limitations that should be addressed in future iterations. The reliance on simulation tools like GNS3, rather than physical equipment, may limit the system's ability to replicate real-world network conditions fully. As underlined by (Muhammad & Munir, 2023) testing automation solutions on physical hardware is essential for ensuring their effectiveness in production environments. Also, the system's focus on a limited set

of network devices and lack of comprehensive network inventory and cybersecurity hardening features suggest room for expansion to meet the broader needs of enterprise networks.

The evaluation and testing phase of the system, which included both functionality and configuration time testing, provides valuable insights into its performance. The successful automation of VLAN configuration, backup, and restoration across multiple devices demonstrates the system's potential to reduce manual workload and improve operational efficiency significantly. This aligns with industry trends toward more comprehensive network automation solutions that can handle diverse tasks across heterogeneous network environments (Santyadiputra et al., 2021).

2.9 Conceptual Framework

The conceptual framework of the proposed system is structured around Django's Model-View-Template (MVT) architecture, with each component playing a specific role to bridge the gaps identified in manual network management. At the core, the urls.py file acts as the gateway for all user requests, mapping URLs to the appropriate view functions and ensuring efficient routing of web traffic. The views.py module handles these web requests and responses, orchestrating CRUD (Create, Read, Update, Delete) operations and serving as the intermediary between the user interface and the backend logic.

The models.py file defines the structure of the database using SQLite, enabling robust storage, retrieval, and management of network inventory, configuration data, and cybersecurity records. Background processes such as equipment login, scheduled network scans are managed by tasks.py, which ensures that routine operations are executed seamlessly without manual intervention. Interfacing directly with network devices, task.py executes device-specific commands and collects operational data, thus closing the loop between the system and the physical network infrastructure. These components are tightly integrated, allowing for automated workflows, centralized data management, and real-time device interaction, all accessible via a user-friendly web interface that leverages modern web technologies. This cohesive design not only streamlines network inventory and configuration management but also enhances cybersecurity oversight, ultimately providing a scalable and collaborative platform that addresses the inefficiencies of manual network management systems. Applications such as network management and configuration, network audit and compliance, and user management all incorporate the files urls.py, views.py, tasks.py and models.py, to ensure smooth and efficient

operation as well as modularize the system for easier troubleshooting. Figure 2.16 visually illustrates how these components interoperate to deliver a unified, efficient, and automated network management solution (Kumar & Nandal, 2024).

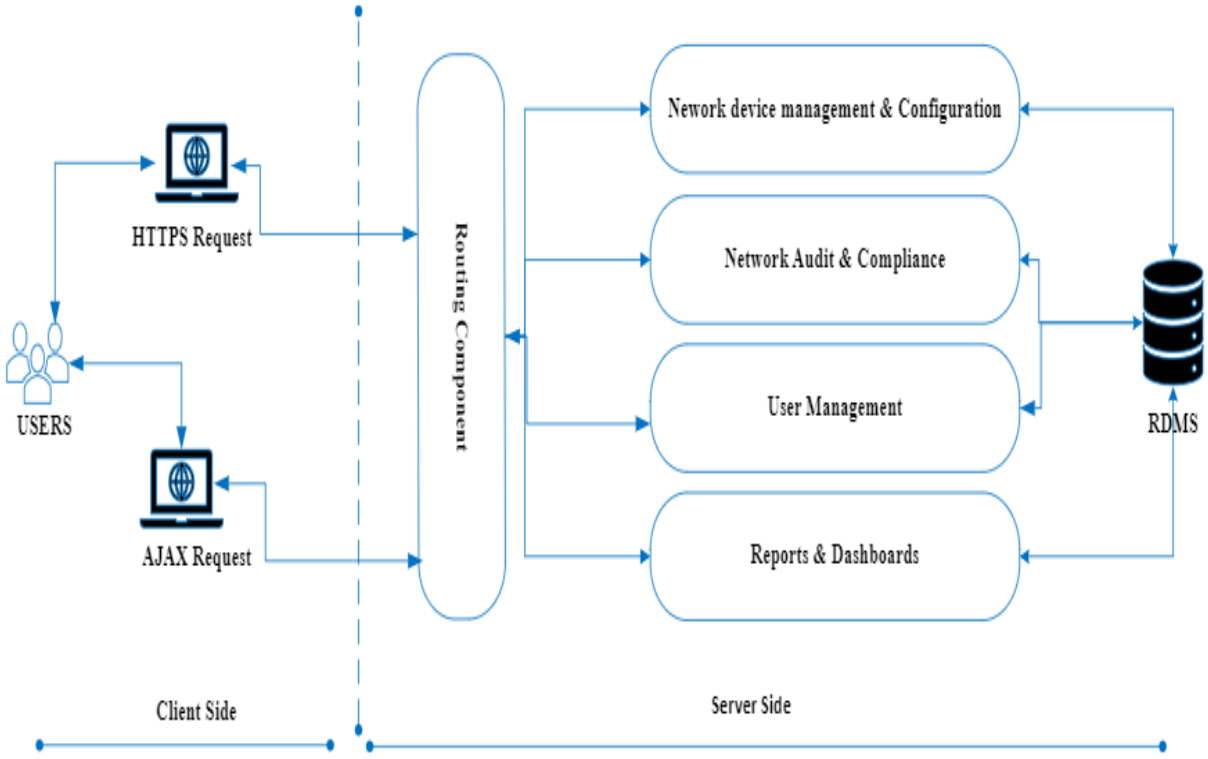


Figure 2.16:Block Diagram of the proposed System

2.9.1 Service Provider Network Components

The service provider network comprises several critical components: the access network, metro network infrastructure, aggregation network, core network, and data center. Each section is vital in delivering connectivity and ensuring high performance and reliability. The access network connects end-users to core services using last-mile and wireless technologies, emphasizing cybersecurity and redundance. The metro network provides high-capacity connectivity across metropolitan areas, efficiently aggregating traffic. The aggregation network consolidates traffic from the access layer, optimizing performance and scalability. The core

network serves as the main pathway for data traffic across long distances, ensuring high availability with advanced technologies like internet Protocol multiprotocol switching (IP/MPLS) and border gateway Protocol BGP. Finally, the data center offers a centralized facility for managing servers and networking equipment, essential for operational efficiency. This comprehensive system effectively meets the intricate demands of large service provider networks, ensuring seamless end-to-end network management and configuration (Brown et al., 2024).

2.10 Conclusion

Network automation tools have fundamentally transformed configuration management, enabling organizations to streamline operations, reduce manual errors, and improve scalability. Tools like Ansible, Salt, Puppet, and Chef each offer distinct approaches tailored to various operational needs. Ansible is widely recognized for its agentless architecture, which simplifies deployment and cross-vendor automation using YAML-based playbooks, though its reliance on YAML can present a barrier for non-programmers (Katariya et al., 2025). In contrast, Puppet and Chef employ agent-based models that enforce configuration consistency and policy compliance across large-scale environments, but their declarative languages and resource-intensive agents can pose challenges for new users and smaller teams (Özdoğan et al., 2023). Salt stands out for its hybrid flexibility, supporting both agent-based and agentless deployments and offering event-driven automation for real-time responsiveness, albeit with a complex master-minion architecture that may overwhelm less experienced teams (Katariya et al., 2025).

Beyond these core configuration tools, specialized solutions such as Scrapli, Netmiko, NAPALM, Nornir, and FastAPI address more granular network management tasks. Scrapli and Netmiko facilitate direct device interaction over SSH or Telnet, providing granular control for scripting but requiring Python proficiency and lacking native idempotency (Osei-Wusu et al., 2025). NAPALM delivers a unified API for managing multi-vendor configurations and state validation, though its dependence on device APIs can limit compatibility with legacy systems (Stancu et al., 2019). Nornir, as a Python-native automation framework, excels in complex, customizable workflows but demands programming expertise and additional effort to ensure idempotency. FastAPI, while not a network automation tool per se, is increasingly used to develop interactive web-based management interfaces due to its high performance, automatic

documentation, and robust data validation, though its Python-centric design and asynchronous programming model may introduce a learning curve (Muhammad & Munir, 2023).

Selecting the right combination of these tools is crucial and should be guided by the organization's network requirements, team skillsets, and the need for scalability and integration. While each tool brings unique strengths, their limitations ranging from steep learning curves to integration challenges highlight the importance of a strategic, context-aware approach to network automation (Clemm et al., 2022). Ultimately, leveraging the complementary features of these platforms can drive operational efficiency and foster innovation in network management.

3: Methodology

3.1 Introduction

An organized research strategy was crucial, for developing studies with a structure. According to the National Center for Biotechnology Information (NCBI) many research inquiries tended to repeat themselves, highlighting the need for a systematic approach to efficiently guide researchers through the process. This systematic method ensured focus and coherence, making the study more manageable and structured. It was essential to consider factors such as research questions, the relevance and limitations of research objectives, feasibility, and existing methodologies, in the literature when selecting a research methodology. Addressing these aspects helped align the chosen methodology with the study objectives and real-world constraints, enhancing the researcher's credibility and impact. Researchers can enrich their fields by selecting methodologies and following processes that yielded grounded results (Oppliger et al., 2022). The following factors are recommended when selecting the most appropriate methodology for each research:

i. What were the existing methodologies applied in the currently current literature?
ii. Practicality, relevance, and restrictions.
iii. Research characteristics – aims and research queries, extent, and so forth.

3.2 Research Setting

This empirical study was conducted within the technology division of Safaricom Kenya limited, a leading service provider in the telecommunications sector headquartered in Nairobi, Kenya. The organization was selected due to its extensive, complex network infrastructure and its ongoing commitment to digital transformation and automation initiatives. With a workforce of over 200 network engineers and related technical staff, Safaricom provided a robust environment for examining the challenges and opportunities associated with automating network management processes. The choice of this organization was informed by its representative scale, diversity of network operations, and the availability of a wide range of network management practices and technologies. Additionally, the organization's strategic focus on operational efficiency made it an ideal setting for evaluating the effectiveness and impact of the proposed automated network management system. Conducting the study in this context ensured that the findings would be both practically relevant and generalizable to similar large-scale service

provider environments. The system design and requirements are based on data Safaricom Kenya Limited which may limit generalizability. Adaptation may be required for other organizations.

3.3 Research Design

The research design for positivism philosophy was comprised of a structured and objective view of data collection and analysis. Positivist researchers used quantifiable observations and statistical analyses to generate factual knowledge, grounded in the belief that reality was independent of the observer and could be measured objectively (Dudovskiy, 2023). The philosophy demanded the use of highly structured methodologies, such as experiments, surveys, and structured questionnaires, to ensure replicability and generalizability of findings (Rashid, 2023). Following the scientific method with prioritized empirical observation and deductive reasoning, positivist research focused on uncovering causal relationships and predicting events, thereby contributing to a more precise and reliable understanding of the social world (Mohammad, 2024).This approach not only ensured the reliability of the research but also ensured that the findings were free from personal biases and subjective interpretations, making them highly valuable for policy-making and practical applications.

3.4 Research Philosophy

In this study, positivism was applied by focusing on objective, measurable data collected through structured surveys, quantitative metrics, and systematic observation of network automation tools. The researcher-maintained neutrality by using standardized instruments and statistical analysis, ensuring that findings were based on factual evidence rather than personal opinions. This approach allowed for empirical testing of hypotheses and produced results that are observable, replicable, and generalizable to similar network management contexts (Dudovskiy, 2023).

3.4.1 Application of Positivism in This Study

a) Empirical Observation

i. Network logs: Automated system logs from Safaricom Kenya's network infrastructure (e.g., configuration changes, downtime records) were analyzed to quantify manual vs. automated task efficiency.

ii. Structured surveys: Engineers were asked to respond to specific automation challenges (e.g., "What are the pain points in there day to day network operations?") to generate measurable data.

b) Objective Measurement

i. Key metrics: Time-to-deploy services, frequency of configuration errors, and downtime duration were tracked using Safaricom's network monitoring tools (e.g., SolarWinds,).

ii. Quantitative focus: All variables (e.g., "What specific automation tools API or frameworks are implemented in the engineers network management processes ") were translated into numerical values for statistical analysis.

c) Hypothesis Testing

i. Hypothesis: "A Django-based automation system reduces service deployment time significantly compared to manual methods."

ii. Testing: Pre- and post-implementation metrics were compared.

d) Generalizability

i. Stratified sampling: Engineers were grouped by role (core network, field engineers, etc.) to ensure findings apply across Safaricom's diverse teams.

e) Value Neutrality

i. Bias mitigation: Surveys used google-forms questions to minimize subjective interpretations. Data was anonymized to reduce respondent bias.

3.5 Population and Sampling

The technology division of the company and affiliates for which this dissertation was created comprised more than 200 network engineers. A significant advantage of employing stratified sampling was that it guaranteed diversity within the sample (Thomas, 2023).

When determining the sample size, several factors were considered:

i. Confidence interval: This referred to the margin of error, which measured the degree of uncertainty or certainty. It indicated the level of confidence one could have in the study results (Kibuacha, 2021).

ii. Confidence level: This was the probability percentage that the confidence interval would contain the true population parameter when a random sample was drawn multiple times (Kibuacha, 2021).

iii. Standard deviation: This statistic helped estimate how much the predictions will be varied from each other and from the average.

To calculate the sample size, the Andrew Fisher's function was used :

$$\text{Sample Size} = \frac{(\text{Z-score})^2 \times \text{StdDev} \times (1-\text{StdDev})}{(\text{confidence interval})^2}$$

Figure 3.1:How to calculate sample size (Kibuacha, 2021)

Where StdDev was the standard deviation, and the z-score was calculated from the confidence level. As the researcher planned to use a confidence level of about 90%, the following sample size was determined based on a z-sore of 1.44, a confidence interval of 10 and a standard deviation of 0.5 (a safe choice since the size of the Service Provider Networks from which data would be collected was unknown). Substituting the corresponding values, then the expected sample size was 10 Network teams.

3.5.1 Sampling Procedure

The sampling procedure for this study was designed to ensure a representative and diverse sample of network engineers and related professionals within the technology division of Safaricom Kenya. The sampling frame consisted of a comprehensive list of all network engineers, system administrators, and field engineers employed in the division, totaling over 200 individuals. To capture the heterogeneity of roles and expertise, the population was first divided into distinct strata based on job function (e.g., core network engineers, access network engineers, system administrators, and field engineers). Each stratum was defined to be mutually exclusive and collectively exhaustive, ensuring every eligible participant was included in one and only one group (Khan et al., 2023).

A stratified random sampling technique was employed, where the proportion of participants selected from each stratum reflected their actual distribution within the organization (McCombes, 2019). For example, if core network engineers comprised 40% of the total

population, they made up approximately 40% of the sample. Within each stratum, participants were selected using a random number generator to minimize selection bias and enhance the representativeness of the sample.

Participant selection criteria included current employment in the technology division, direct involvement in network management or operations, and at least one year of experience in the organization. Exclusion criteria were contractors, interns, and those not directly involved in network management. This approach ensured that the sample included a broad spectrum of expertise and perspectives relevant to the research objectives (Bisht, 2023).

The final sample size was determined using standard formulas for confidence interval and confidence level, as described in Section 3.4, resulting in a target of 10 network teams. This stratified approach not only improved the accuracy and generalizability of the findings but also ensured that all key subgroups were adequately represented in the analysis (Ahmed, 2024).

3.5.2 Sampling variables

The variables for this study were defined to align with the research objective of evaluating the impact of network automation in a service provider environment.

a) Independent Variable:
    i. Level of network automation (measured as the proportion of automated versus manual network management tasks).
b) Dependent Variables:
    i. Service deployment time (minutes/hours required per deployment).
    ii. Frequency of network configuration errors (number of errors per deployment).
    iii. Network downtime (minutes/hours per month).
    iv. Engineer workload (number of manual configuration interventions per month).
    v. Engineer satisfaction with network management processes (measured on tasks completion efficiency and accuracy).
c) Control Variables:
    i. Type of network device (e.g., routers, switches).
    ii. Team or department.

iii. Complexity of service deployed.

These variables were operationalized through structured questionnaires and system logs, ensuring that all data collected could be analyzed quantitatively. The explicit identification and definition of these variables guided both the data collection and analysis processes, ensuring methodological rigor and alignment with the study's positivist philosophy (Kentik, 2025; Muhammad & Munir, 2023).

3.6 Analysis

3.6.1 Data Collection

Data was gathered directly from network engineers, system administrators, field engineers and other related network roles. The primary benefit of collecting data directly from the source was that it was more reliable and precise, leading to better results and solutions. Another advantage of gathering primary data in this manner was the reduced costs, as there was no requirement for professional researchers or investment in extra equipment (Mohammed, 2024).

Another reason for opting for this collection model was to obtain accurate information regarding the functionality of the service provider network ecosystem. Questionnaires were generated and administered through Google Forms. This approach allowed the researcher to access details from engineer operating the service provider network infrastructures. Questionnaires enabled the researchers to obtain objective and subjective data since different respondents answer similar questions. As such, it was easier to identify areas where the majority agree or disagree. Thus, the researcher can observe common responses to determine the data patterns (Taherdoost, 2021).

(Maptionnaire, 2022) recommended that the following seven steps and guidelines be followed when designing a questionnaire:

i. Identify what needed to be covered in the questionnaire.
ii. Avoid mincing words.
iii. Ask questions one at a time.
iv. Provided options in questions that were flexible enough to cover all possible outcomes.

v. Carefully decided which questions should be open ended and which should be close ended.
vi. Know the audience, as it determines the languages used and the structure of the questions.

Choose the right tool how the questionnaires would reach respondents (online forms, email, etc.)

3.6.2 Data Validation

Upon completing the data collection exercise, the researcher was solely responsible for validating the data to ensure its accuracy, completeness, and adherence to scientific research standards. The validation process involved a systematic review of all collected questionnaires and responses, checking for missing values, inconsistencies, and outliers (Cobern & Adams, 2020). The researcher cross-verified responses for logical consistency and ensured that all entries met the predefined inclusion criteria for the study. Where clarification was required, the researcher directly contacted participants by email or phone to confirm the authenticity of responses and resolve any ambiguities, always maintaining the confidentiality and privacy of respondents in accordance with ethical guidelines (Elsevier, 2022). Specific data validation checks included ensuring all required questions were answered, screening for duplicate or fraudulent entries, verifying those responses aligned with the sampling and research criteria, and reviewing data for logical consistency and completeness. Any records with unresolved discrepancies were excluded from analysis to maintain data quality and integrity. The entire validation process was documented for transparency and auditability, in line with best practices for empirical research (Gottfried, 2024).

3.6.3 Data Analysis

The data gathered for research purposes consisted of structured and quantitative information. Quantitative data analysis involved examining numerical data or data that could be readily transformed into numbers without losing its significance (Rahman & Muktadir, 2021). Quantitative analysis was generally used for:

i. Evaluating the differences among groups.
ii. Examining the relationships between variables.
iii. Testing hypotheses in a scientifically rigorous manner where two main steps were involved quantitative data analysis:

i. Data Preparation

Data preparation involved converting the raw data collected into meaningful and readable information. It was prepared in the following two steps:

a) Data validation ensured that the data collection was conducted according to set standards and without any prejudice. The purpose of data validation was to find out, as far as possible, whether the data collection was done as per the pre-set standards and without any bias. It was a four-step process, which included:
    i. Fraud detection, to infer whether each respondent was genuinely interviewed.
    ii. Screening to ensure that respondents were chosen according to the research criteria.
    iii. Procedure verification: to check whether the data collection procedure was duly followed.
    iv. Completeness check, to ensure that the interviewer asked all required questions rather than just a few.
b) Data editing: aimed at ensuring there were no errors in the data. This process involved conducting basic data checks, identifying outliers, and potentially removing data points that could affect the accuracy of the results.
c) Data coding - involved grouping data and assigning values to these groups for ease of analysis.

ii. Data analysis

After the data was validated, the actual analysis began. The researcher focused on both descriptive and inferential statistical methods. Descriptive analysis aimed to summarize the data and identify patterns, such as the range of unautomated service provider network services, the mean of automated services, and the frequency with which engineers used automated systems. Inferential analysis was used to examine relationships between variables and to test hypotheses, allowing for generalization of findings and prediction of trends within the population. All analyses were conducted using objective, numerical methods appropriate for quantitative research (Rahman & Muktadir, 2021).

3.7 System Development Methodology

The system was developed employing an Agile approach. Agile is an iterative method for software development and project management that enables teams to deliver results more quickly and consistently. It establishes a common foundation among various frameworks, including Scrum, Extreme Programming, and Crystal Clear. The project adhered to the Agile Software Development Manifesto, which is grounded in four fundamentals.

i. Prioritizing individuals and interactions over processes and tools.
ii. Emphasizing working software rather than extensive documentation.
iii. Fostering customer collaboration instead of focusing solely on contract negotiation.
iv. Valuing adaptability to change more than strictly adhering to a plan.

The primary reason for adopting the Agile methodology was its reputation for enhancing the success of software projects in addressing user, customer, and business needs, while also enabling faster and more responsive software delivery compared to traditional waterfall methods (Daraojimba et al., 2024).

3.7.1 Agile Development Lifecycle

According to (Landau, 2022) the Agile philosophy is founded on 12 essential principles:

i. Customer satisfaction achieved through early and continuous delivery of software.
ii. Adaptability to changing requirements throughout the development process.
iii. Frequent delivery of functional software.
iv. Collaboration between business stakeholders and developers during the entire project.
v. Support, trust, and motivation for all participants involved.
vi. Facilitating face-to-face communication.
vii. Working software serves as the primary indicator of progress.
viii. Agile processes promote a sustainable development pace.
ix. Attention to technical details and design improves agility.
x. Emphasis on simplicity.
xi. Self-organizing teams foster better architectures, requirements, and designs.
xii. Regularly reflecting on ways to enhance effectiveness.

Figure 3.2:Agile Model (Milne, 2022)

3.7.2 Agile Development Phases and Key Activities

This system was developed using the Agile Software Development Life Cycle (SDLC), which emphasizes iterative progress, collaboration, adaptability, and continuous feedback. The Agile SDLC for this project was structured into six key phases: Concept, Inception, Iteration (Development), Testing, Deployment (Release), and Review/Maintenance (DiCesare, 2025).

i. Concept (Ideation):

In this initial phase, the project vision and objectives were established. The product owner and stakeholders collaboratively defined the scope, identified the primary challenges faced by network engineers, and determined the desired outcomes of the automated network management system. Key activities included stakeholder interviews, requirements gathering, and feasibility analysis to ensure the project addressed real-world needs.

ii. Inception (Planning):

Detailed planning was conducted, including the creation of the product backlog, prioritization of features, and estimation of resources and timelines. User stories were developed to capture specific requirements from network engineers and system administrators. The technical stack (Django, Python) selected and established the initial design framework.

iii. Iteration (Development):

The core development occurred in this phase, structured into multiple sprints. Each sprint focused on delivering specific features or modules, such as the user interface, device inventory management, and automation scripts. Activities included coding, UI/UX design, integration of front-end and back-end components, and regular sprint planning and review meetings. Continuous feedback from network engineers was incorporated to refine requirements and ensure the solution remained aligned with user needs.

iv. Testing:

After each iteration, developed features underwent rigorous testing. This included unit testing, integration testing, and user acceptance testing. Network engineers participated in testing to validate functionality, usability, and performance. Bugs and issues identified were documented and addressed in subsequent sprints, ensuring high software quality and reliability.

v. Deployment (Release):

Once features passed testing, they were deployed to the test environment. The deployment phase included final system integration, data migration, and user training. Documentation was prepared, and support resources were provided to facilitate adoption by network engineers. Feedback from initial deployment was collected to guide further improvements.

vi. Review/Maintenance:

After deployment, the system entered the review and maintenance phase. User feedback was collected, and updates implemented, enhancements was done. Regular review meetings ensured that the system continued to meet evolving requirements and maintained optimal performance.

3.7.3 Rationale for Agile Adoption

Agile was chosen for its iterative nature, ability to incorporate continuous feedback, and proven effectiveness in delivering user-centered, high-quality software solutions in dynamic

environments. The involvement of network engineers throughout all phases ensured the final system was robust, relevant, and adaptable to future needs (Dugbartey & Kehinde, 2025).

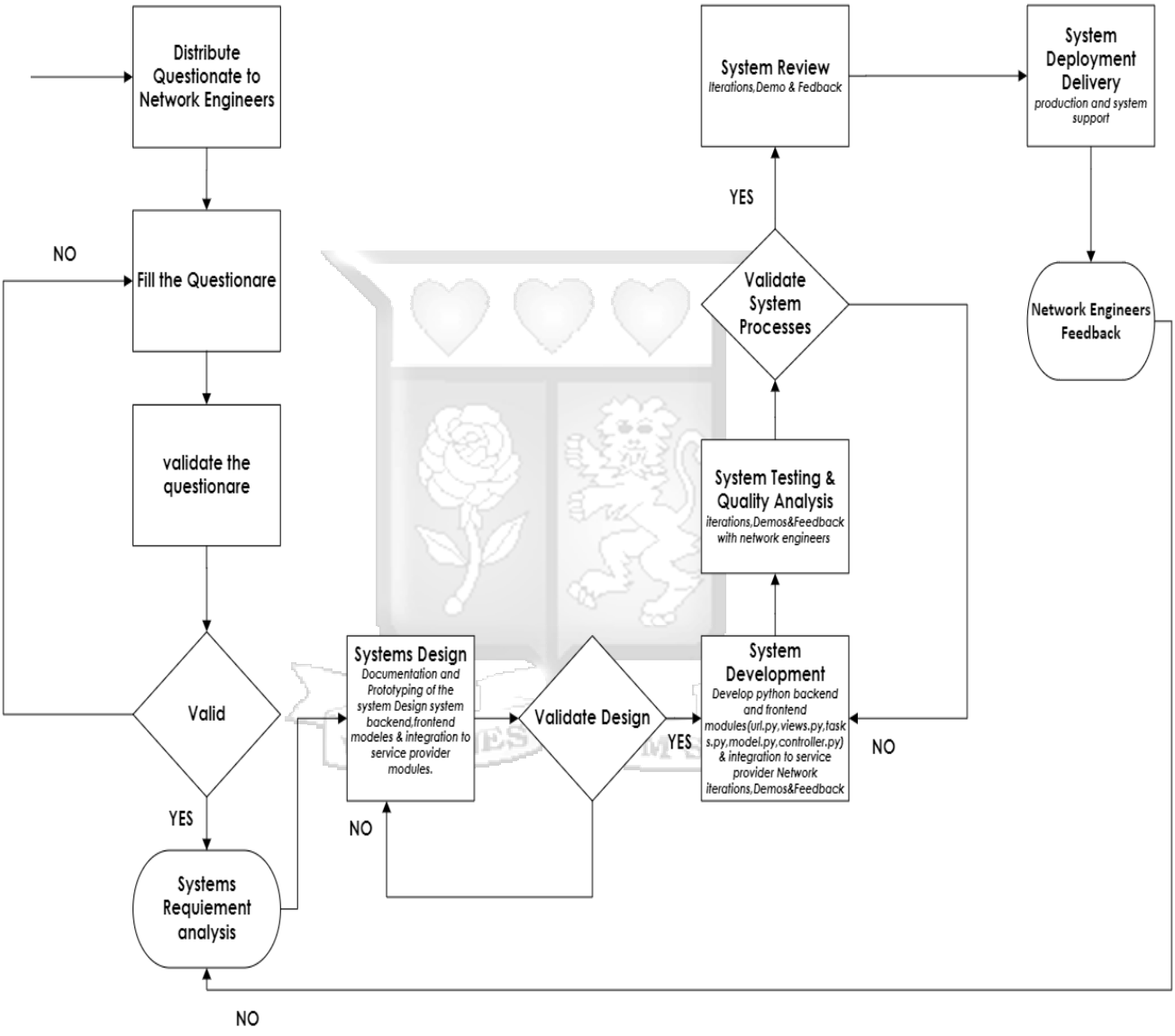


Figure 3.3:System Design Flowchart

3.8 Dissemination of the Results

After developing the system, the developer published a dissertation highlighting the problem, proposed solution, and results. The publication of the research findings was essential in

detailing the scientific and practical contributions of the system. The publication enables researchers and practitioners who are interested in the system to advance their knowledge or improve upon it (Liu et al., 2022).

Reports and proof-of-concept models were also presented to service providers network engineers.

These reports may include:

i. The average time required to deploy a service on the network.
ii. Cybersecurity compliance and remediation recommendations for the network.
iii. Network inventory information.
iv. At a class network dashboard with relevant KPI's
v. Discrepancies between manual and automated network processes.

The goal is to provide valuable insights to service providers that can inform their decision-making and help optimize network operations.

3.9 Ethical Considerations

The researcher upheld all ethical guidelines during and after collecting essential data. The primary obligations were to maintain confidentiality and privacy of the data provided by the respondents. The researcher emphasized that the information was kept confidential and used solely to complete the current research. Electronic records were secured in a locked compartment where unauthorized persons could not access them. According to the principle of beneficence, during the research process, the researcher ensured the safety and welfare of the respondents (Infonetica, 2024).

4: Results and Analysis

4.1 Introduction

Data was collected and then processed in answer to the problems discussed in chapter one of this dissertation. The fundamental objectives guided the collection of the data and consequent data analysis. The objective was to determine the current problems facing network management for service provider network architectures and the problems faced by network engineers in their day-to-day network operations. Further, it was to determine if current network management procedures provide adequate functionality to manage key network functions. These objectives were accomplished fully. The results presented in this section show the prospective service providers network not only solving the problems highlighted in the problem statement but to goes a step further to improve network efficiency, while taking into consideration of new network features like network automation.

All data presented in this chapter was collected Safaricom Kenyan Limited. As such, the findings may not be representative of all service providers in Kenya, other African countries, or globally. The results reflect the specific practices and perceptions within this organization.

4.2 Limitation of the data

It is important to note that the data presented in this chapter was collected exclusively from a single service provider in Kenya (Safaricom Kenya Limited). As such, these findings may not be representative of all service providers in Kenya, other African countries, or global contexts. The results reflect the specific practices, perceptions, and challenges within this organization. Additionally, the study relied on self-reported survey responses rather than direct technical audits or observations of manual and automated processes. Terms such as “somewhat manual” and “somewhat automated” are subjective and may overlap, meaning the data may not precisely reflect the actual level of automation in practice, but rather the respondents’ perceptions. Due to resource and access constraints, a comprehensive technical assessment of network processes was not possible. In real-world scenarios, designing an effective network automation solution would typically require more in-depth technical investigation and validation beyond survey responses alone. These limitations should be considered when interpreting the findings and their broader applicability.

4.3 Data Analysis Tools

Analysis was done mainly through windows office excel tables which has rich feature set to come up with the charts and tables. This was after manipulating the data results that had been captured on google forms and entered manually on the excel sheets. Data was taken from online questionnaire response that had been issued online to a select number of the network engineers, systems engineer, field engineers and other network related roles.

4.4 Questionnaire Response Rate

One hundred twenty network engineers out of nearly two hundred were nominated for interviewing based on their distribution around the network engineering teams. Data was collected by filling a simple questionnaire (annexed in the appendix) to capture the information from the network engineers to ease of problem they encounter while using the current network for service delivery. They also appraised on their thought regarding the implementation of a network management that enhanced easy network monitoring, troubleshooting, network health, visualization and service deployment.

A total of hundred and ten questionnaires were distributed across the selected respondents. Figure 4.1 below shows a breakdown of the response rate as determined by the number of questionnaires that were returned.

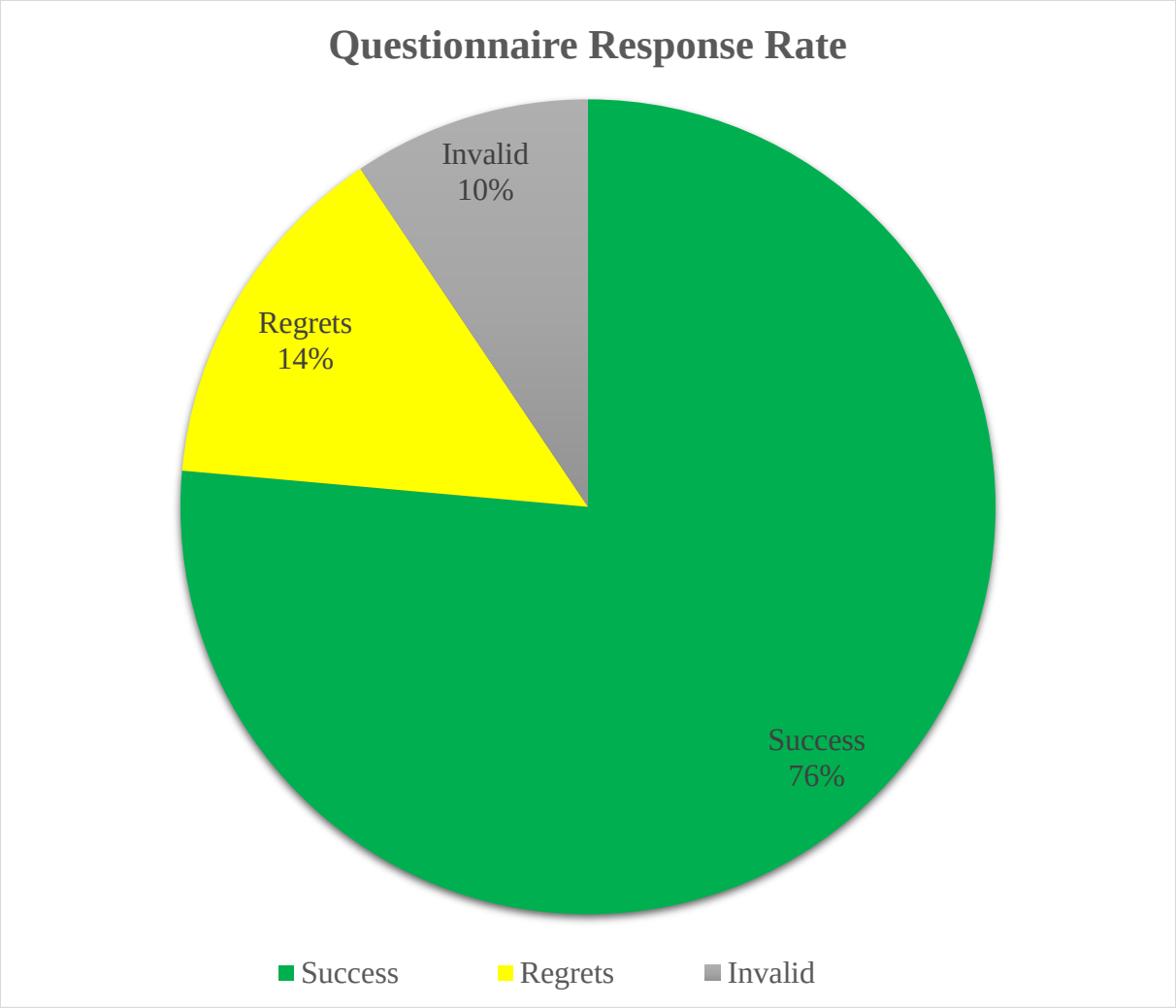


Figure 4.1:Questionnaire Response Rate

4.5 Suitability of the Networks Management

This data indicates that a significant majority, 56% of respondents, rely heavily on manual processes for network management. Only about 35.6% have achieved some automation, with a small number of about and 8 % with full automation. The reliance on manual methods suggests inefficiencies and potential for introduction of vulnerabilities in network operations, emphasizing the need for organizations to transition towards more automated and streamlined network management practices.

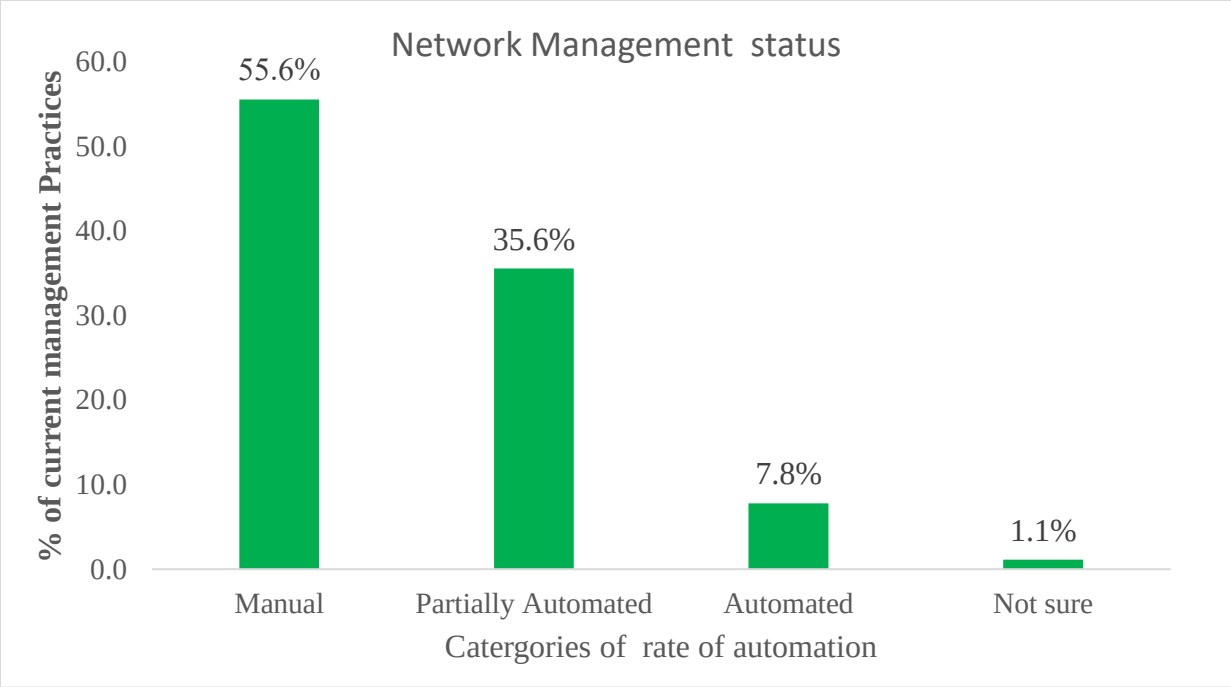


Figure 4.2:Analysis of the Suitability of the Current Network Management

4.6 Network Management Scalability and Cybersecurity

This data suggests that a significant majority of teams are struggling with network management scalability and cybersecurity. Over 40% of respondents indicated their networks are not fully scalable and secured, highlighting a critical gap in network management practices. Only about 33% of respondents reported having networks that are at least somewhat scalable and secured, with less than 11% claiming fully network management scalability and secured. These findings underscore the importance of implementing robust network management strategies that prioritize both scalability and cybersecurity compliance. Service providers should focus on adopting scalable network designs management solutions, implementing advanced cybersecurity measures, and regularly assessing their network infrastructure to address these challenges effectively.

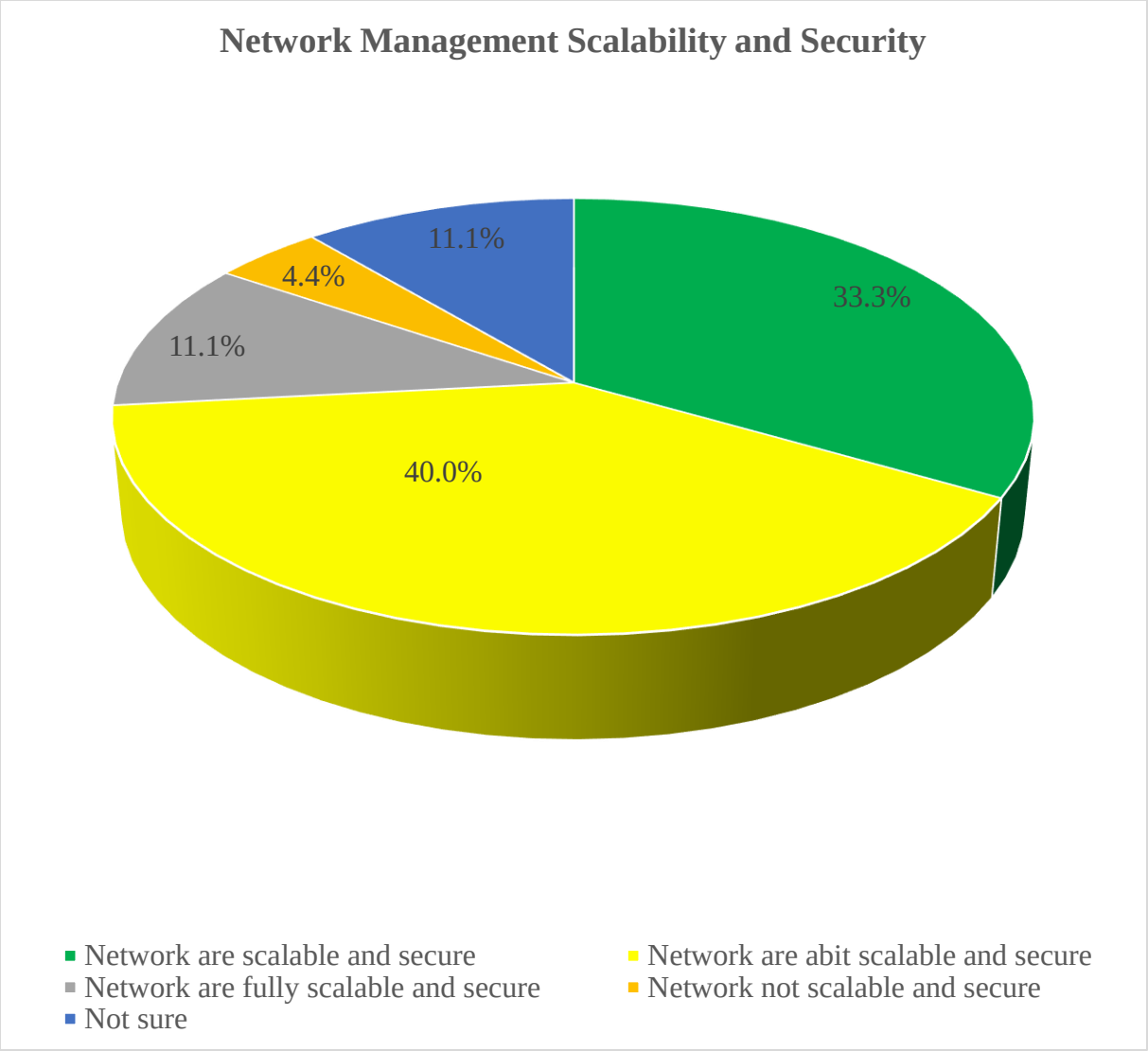


Figure 4.3:Analysis of the Network Management scalability and Cybersecurity

4.7 Tools and Frameworks

The Figure 4.4 below is a percentage breakdown of the existing service provider network automation tools and frameworks. SolarWinds network automation manager led the pack with 34% responses, followed by Red Hat Ansible automation platform with 22%, and other tools not listed had a distribution of about 22%. Other notable tools included chef, saltstack enterprise, Napalm, and Netmiko, each with 5% adoption rate. Nornir and puppet trailed behind with 2 and 1 responses, respectively. About21% of respondents opted for other unspecified tools, reflecting

the diverse needs and preferences within the industry. Overall, the survey demonstrates a strong preference for comprehensive automation solutions.

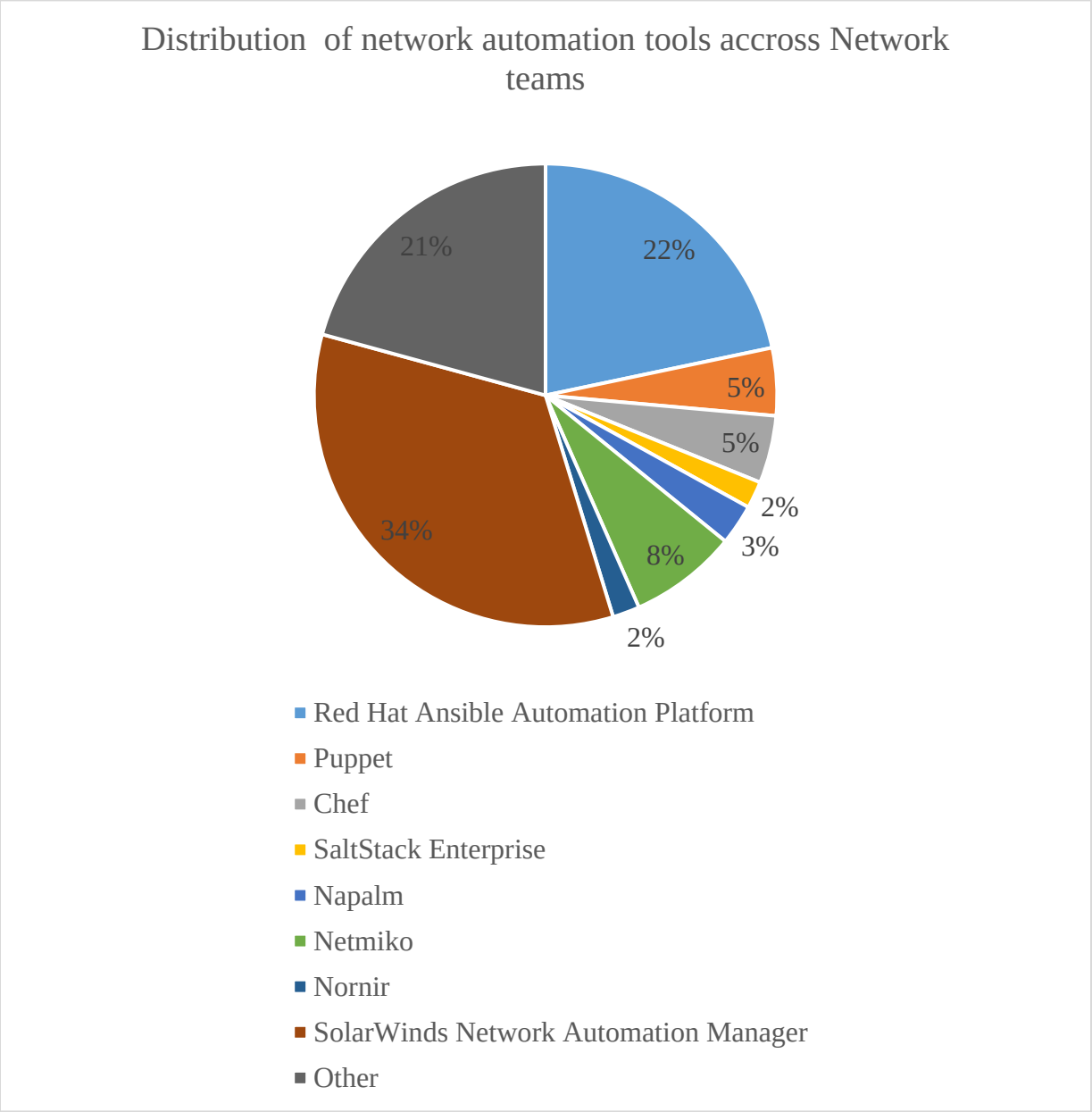


Figure 4.4:Automation tools and Frameworks

4.8 Pain Point on Network Management

Figure 4.5 shows a distribution of various significant pain points in network management. A survey of network engineer's day to day work processes revealed the most pressing challenges they face in their daily work. The overwhelming majority, 25% respondents, cited time-

consuming manual configurations as their primary concern, highlighting the need for more efficient automation processes. Troubleshooting network issues and scaling network infrastructure tied for second place, with 24% responses, underscoring the complexity of modern network environments. The remaining challenges were evenly distributed, with about 12% responses each for keeping up with Cybersecurity threats, managing network performance, maintaining compliance and documentation, and coordinating with other network teams. Interestingly, 3% respondents also mentioned other unspecified challenges, suggesting that the field of network management faces a diverse array of obstacles. This distribution of responses paints a clear picture of the multifaceted nature of network management, with a strong emphasis on the need to streamline service provider network processes and improve overall efficiency.

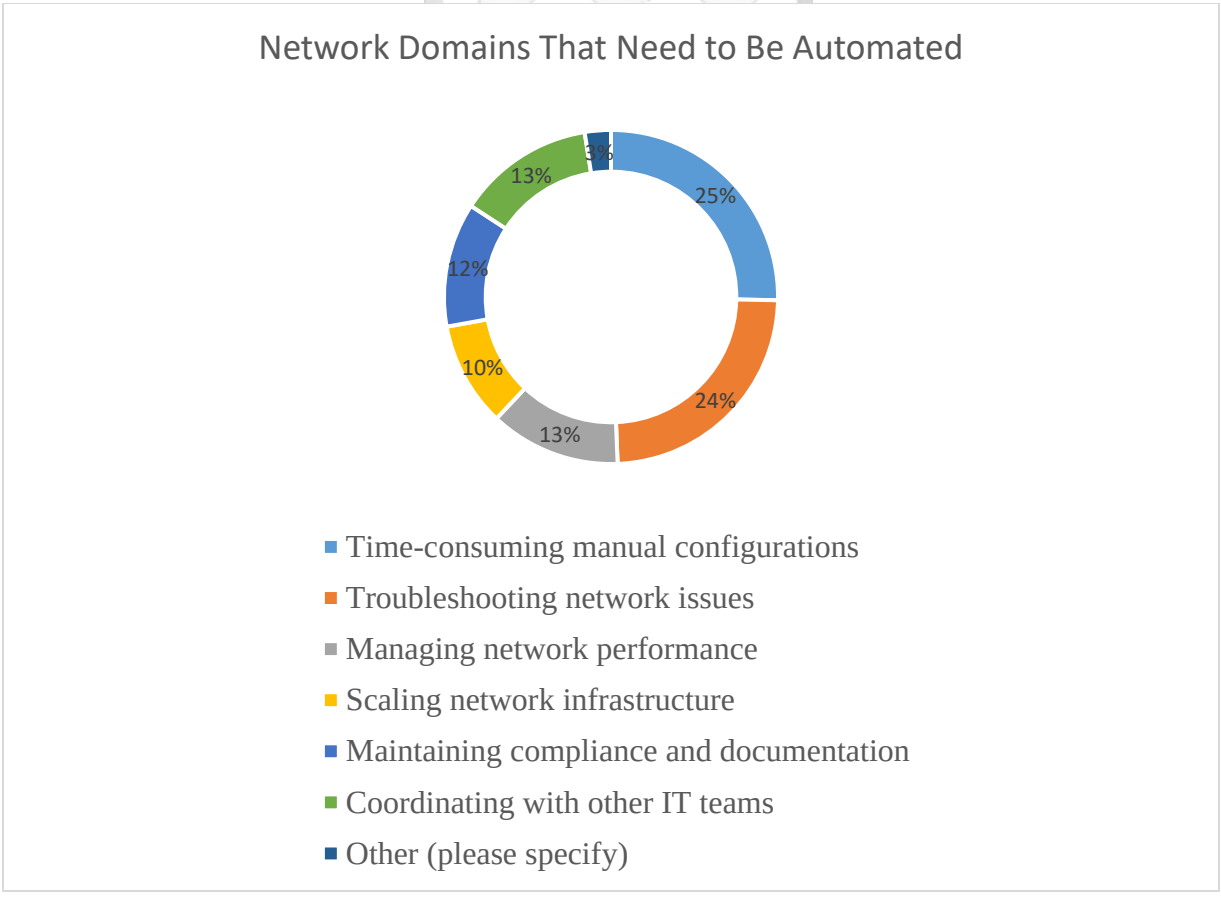


Figure 4.5:Automation tools and Frameworks

4.9 Use-cases to be Automated

Network automation is increasingly vital for organizations seeking to enhance efficiency, reduce errors, and improve overall network performance. The most desired processes for automation include configuration management, device inventory and lifecycle management, compliance and cybersecurity remediation, service provisioning, and network monitoring and alerts. Configuration management tops the list, with a score of 26%, as it ensures consistent and error-free network setups. Device inventory and lifecycle management, with a score of 18%, aiming to streamline device tracking and maintenance. Compliance and cybersecurity remediation, service provisioning, and network monitoring also rank high, with scores of 16%, 19%, and 21% respectively. By automating these network management processes, service providers can significantly reduce burdens, enhance cybersecurity, and improve end-user experiences.

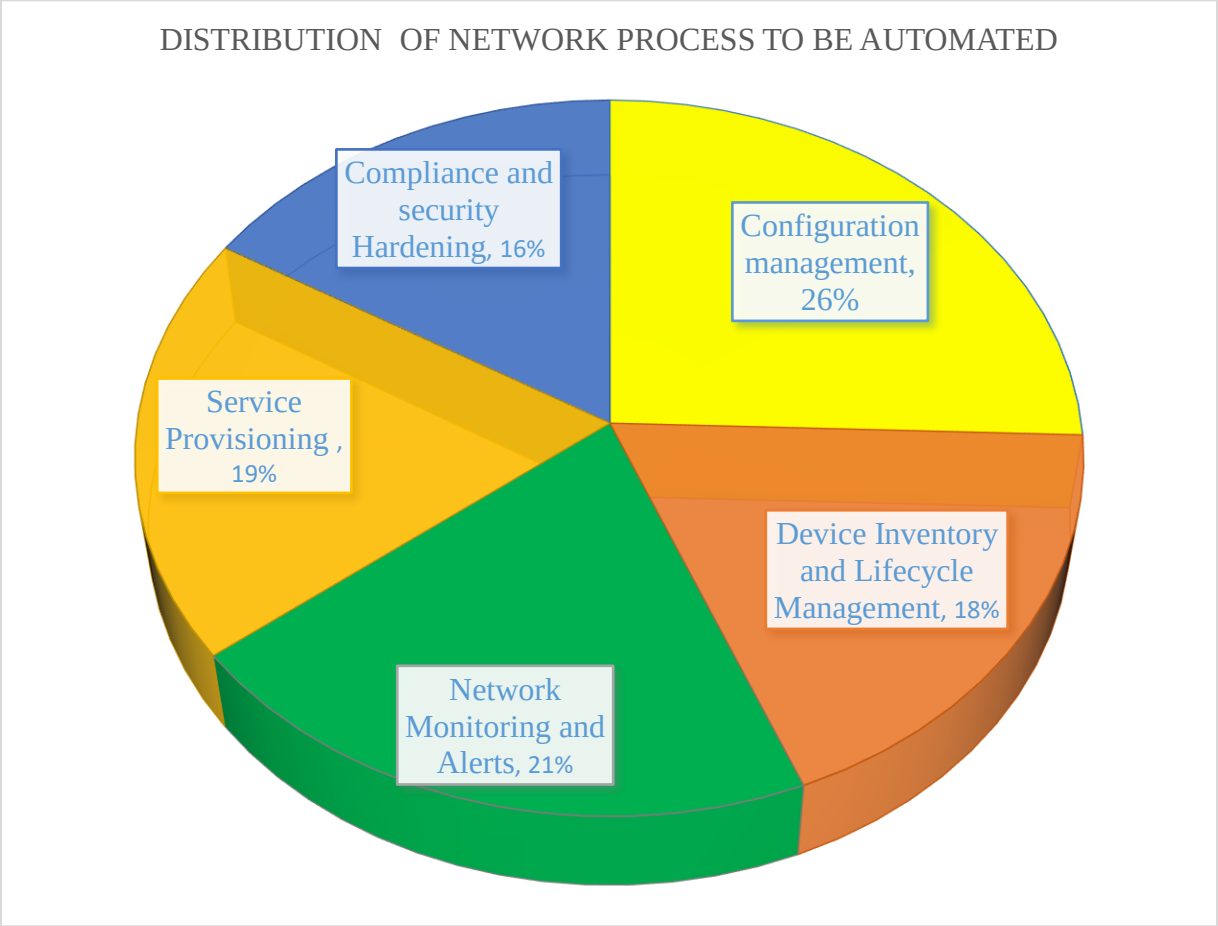


Figure 4.6:Automation tools and Frameworks

4.10 Summary

The current state of network management reveals significant challenges that hinder efficiency and cybersecurity. The survey highlighted that many network engineers rely heavily on manual processes, with only a small fraction achieving full automation. This reliance on manual methods not only leads to inefficiencies but also exposes networks to potential vulnerabilities. Furthermore, the survey showed that close to 40% of respondents struggle with network management and cybersecurity, indicating a critical gap in current network management practices. Only a small percentage reported having networks that are fully scalable and secured, emphasizing the need for robust strategies that prioritize both scalability and cybersecurity.

To address these challenges, network engineers are turning to automation tools and frameworks. SolarWinds Network Automation Manager emerged as the most popular choice, followed by Red Hat Ansible Automation Platform. Other tools like Chef, SaltStack Enterprise, Napalm, and Netmiko also received notable mentions, reflecting the diverse needs within the industry. The primary pain points in network management include time-consuming manual configurations, troubleshooting network issues. To improve efficiency and cybersecurity, there is a strong emphasis on automating key processes such as configuration management, device inventory, cybersecurity compliance service provisioning, and network monitoring. By adopting these automation strategies, service providers can significantly reduce operational burdens, enhance cybersecurity, and improve end-user experiences.

These findings should be interpreted with caution, as they are based on self-reported data from one organization. The limitations outlined above mean that the results may not fully capture the true extent of automation or manual processes, nor are they necessarily generalizable to other contexts.

5: System Design

5.1 Introduction

This chapter presents a comprehensive overview of the design and architecture of the service provider's network management system, developed to address the challenges identified in Chapter 4. The system aims to alleviate the pain points experienced by network engineers in service provider environments, particularly the reliance on manual processes, network inventory, operations and network security concerns. By leveraging modern automation techniques and frameworks, the system is designed to streamline network operations, enhance security posture, and improve overall network efficiency. The design is guided by the principles of modularity, scalability, usability, and security, safeguarding that the system can adapt to evolving network environments and user needs.

Additionally, this chapter presents a network management system design tailored to the specific operational context of Safaricom Kenya Limited, a Kenyan service provider. While the modular architecture provides a foundational framework, the design reflects the unique infrastructure, workflows, and requirements identified in this organization. Implementations in other mobile network operators (MNOs) or regions may require adaptations to accommodate differing technical, regulatory, or operational environments. The design is informed by data from a single firm, and its generalizability is discussed in Chapters 3, 6, and 7.

5.2 System Architecture Overview

The system adopts a modular, three-tier architecture comprising the presentation tier (frontend), application tier (back-end), and data tier. This architecture provides a clear separation of concerns, allowing for independent development, testing, and scaling of each tier. The presentation tier offers a user-friendly interface for network engineers to interact with the system. The application tier handles the core logic, automation workflows, and communication with network devices. The data tier stores network inventory data, administration information, configuration information, and system logs. The three-tier architecture (presentation, application, data) was selected for its modularity and separation of concerns. While the presentation and application tiers are optimized for the workflows and tools used by Safaricom Kenya Limited, the design can be adapted for other organizations with different technical requirements or legacy systems.

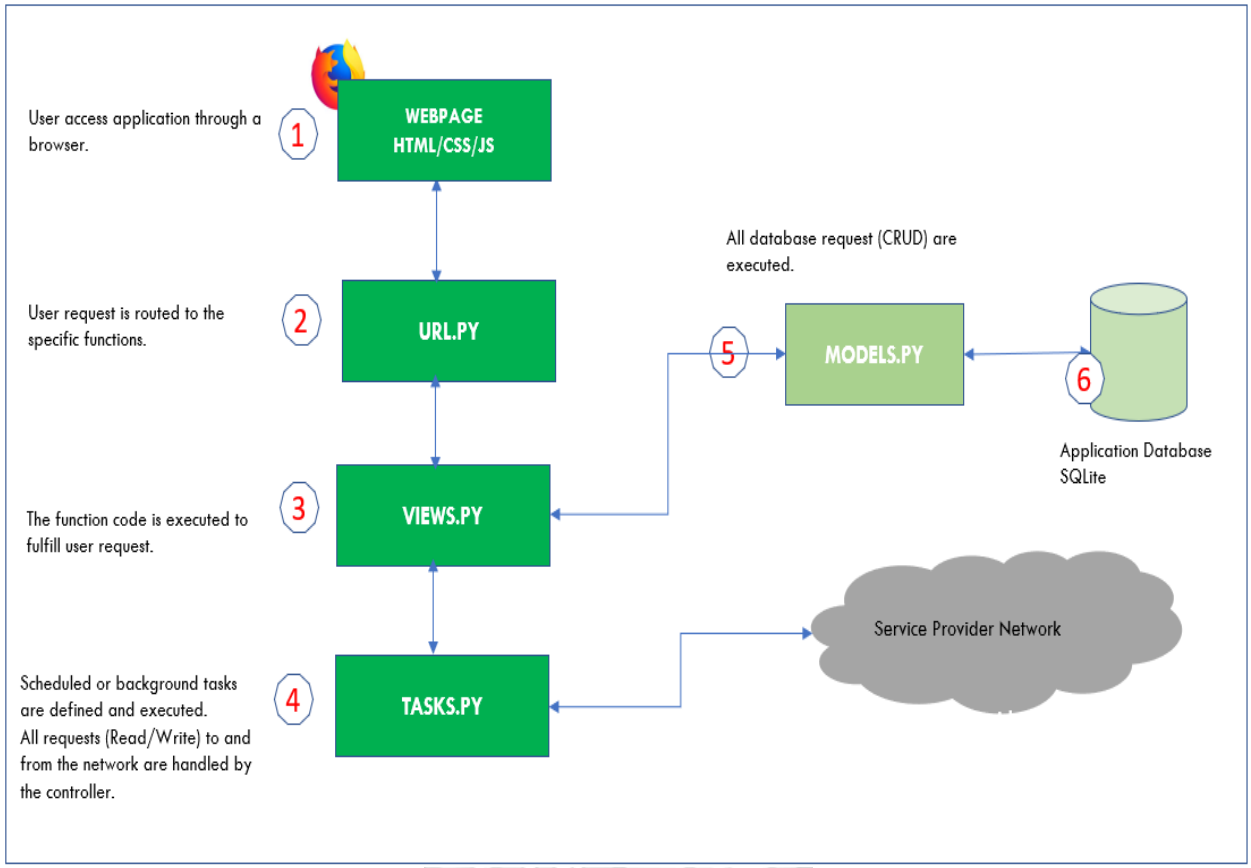


Figure 5.1:Use Case Diagram

5.3 Presentation Tier Front-End

The presentation tier of the network management system built using modern web technologies Django, HTML5, CSS3, and JavaScript, offers a user-friendly interface tailored for network engineers. This tier provides intuitive dashboards displaying critical network status, performance metrics, and cybersecurity alerts, enabling efficient monitoring and decision-making. Key features include views for configuring network devices, provisioning services, and managing security policies, as well as robust search and filtering capabilities for quick access to network inventory data. Additionally, role-based access control ensures that users have appropriate permissions, enhancing security and operational efficiency. This design emphasizes usability and functionality, streamlining complex network management tasks into an accessible and efficient interface.

5.4 Application Tier (Back-End)

The application tier, developed using Python and the Django web framework, forms the core logic of the system. It comprises several key components: RESTful API endpoints for handling requests and device interactions, an automation engine for orchestrating tasks like configuration management and service provisioning, a device interaction module utilizing SSH protocol, a data processing module for data transformation and validation, and a task scheduler for managing scheduled operations like inventory checks. These components work together to provide a robust backend that efficiently handles network management tasks, automates processes, and ensures seamless communication between the presentation tier and network devices.

5.5 Data Tier

The data layer of the network management system utilizes reliable database solution SQLite to act as the primary storage for all system-related data. SQLite's seamless integration with Django and its lightweight design makes it an ideal choice for developers, enabling quick setup and efficient database management (Ramesh et al., 2023). This layer holds vital information, including network inventory (devices, interfaces, and connections), configuration settings (device configurations, security protocols, and service definitions), user accounts (credentials and access levels), system logs (records of activities and errors), and task execution history (logs of automated tasks). By providing centralized storage, this tier facilitates swift data retrieval and effective handling of essential network details, forming a solid base for decision-making and network operations.

5.6 Service Provider Network Architecture

The sample service provider network managed through network automation is illustrated in Figure 5.2. The automation system interfaces with the service provider environment utilizing secure SSH protocol. The network infrastructure comprises various router models.

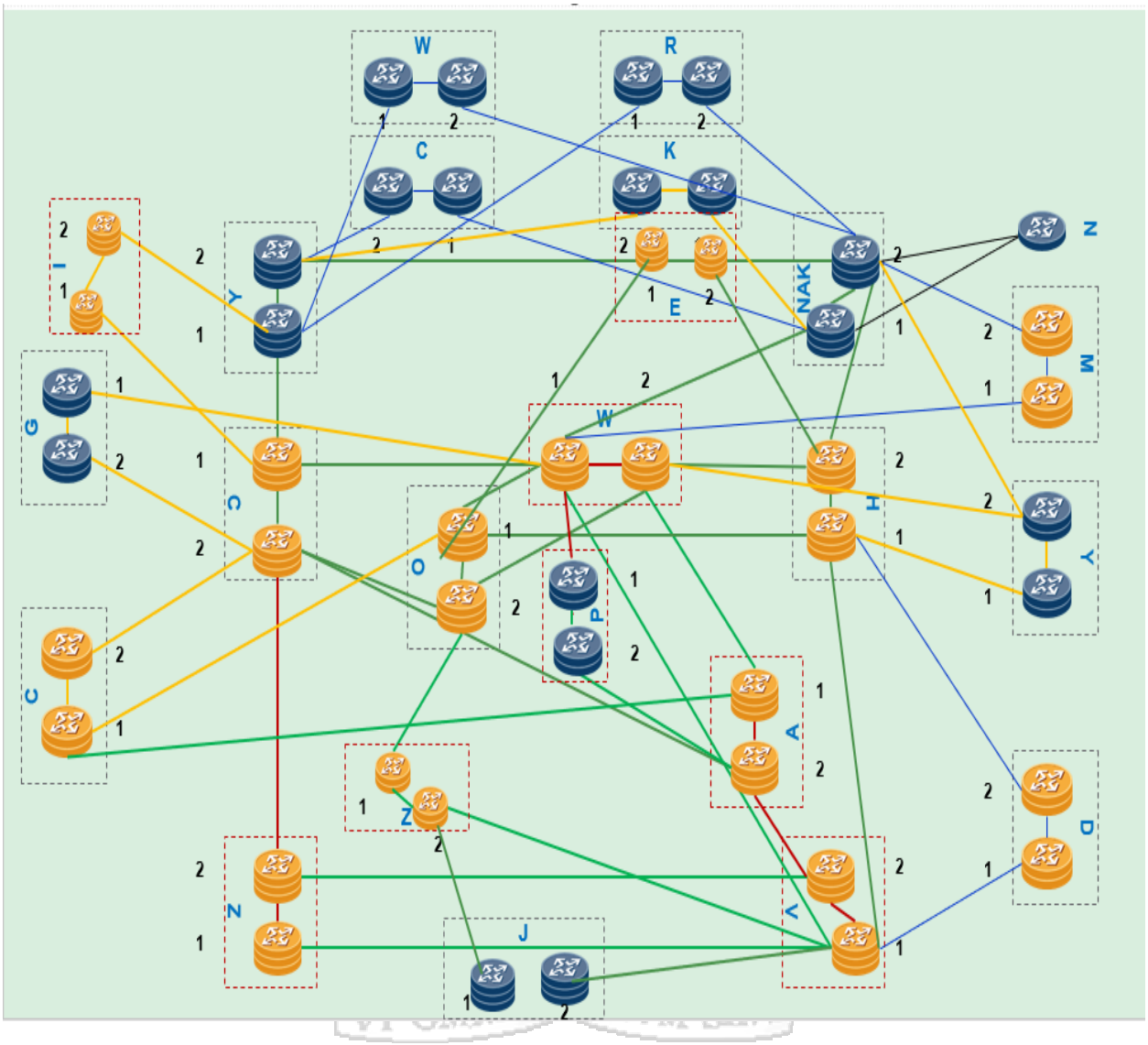


Figure 5.2:Sample Service Provider Topology

5.7 Functional and non-functional Requirements

The functional requirements define what the network automation system must do, including specific tasks and capabilities, such as automation of processes, network data inventory, and service orchestration. Non-functional requirements specify how the system should operate in terms of quality attributes like performance, scalability, reliability, security, maintainability, compatibility, and usability(Saleem et al., 2023; Saroja & Haseena, 2023).

5.7.1 Functional Requirement

i. System should authenticate users.
ii. The system should accurately display a dashboard with accurate network key performance indicators (KPI's).
iii. The system should allow for network report generation.
iv. The system should allow for addition of new network nodes, datacenters.
v. The enforce network compliance.
vi. The system should provide service configuration templates and configure network layer two and layer three services.

5.7.2 Non-functional Requirements

i. The system should provide highly accurate network insights.
ii. Downtime must be minimized, ensuring high availability.
iii. The system should feature a user-friendly and intuitive interface.
iv. Robust security measures must be implemented to protect the model.
v. Network reports should be generated quickly to ensure efficiency.
vi. Service configurations should be performed at optimal speed.

These requirements reflect the operational needs and priorities of the organization studied. Other MNOs may require modifications to accommodate their specific infrastructure, regulatory, or business needs.

5.8 Use Case Diagram

The network automation system use-case diagram figure 5.3 exemplifies the elaborate interactions between actors and system functionalities. It depicts network engineer as primary actors, engaging with key use cases such as "View Dashboard," "compliance checks," " Cybersecurity Compliance," "Update Network Inventory," and "Orchestrate Service Configurations." The other primary actors are the "Network System Administrator" who interacts with functionalities such as "user account management" "monitor the system holistically". The diagram also indicates network manager as the secondary actor who is responsible for basic network performance overview. The use case diagram (Figure 5.3) categorizes each module according to its primary architectural tier: presentation, application, or data. For example, "View Dashboard" and "User Authentication" are part of the presentation tier, while "Orchestrate

Service Configurations" and "Enforce Compliance" belong to the application tier, and "Store Network Inventory" is assigned to the data tier. This mapping illustrates the modular structure and separation of responsibilities within the system.

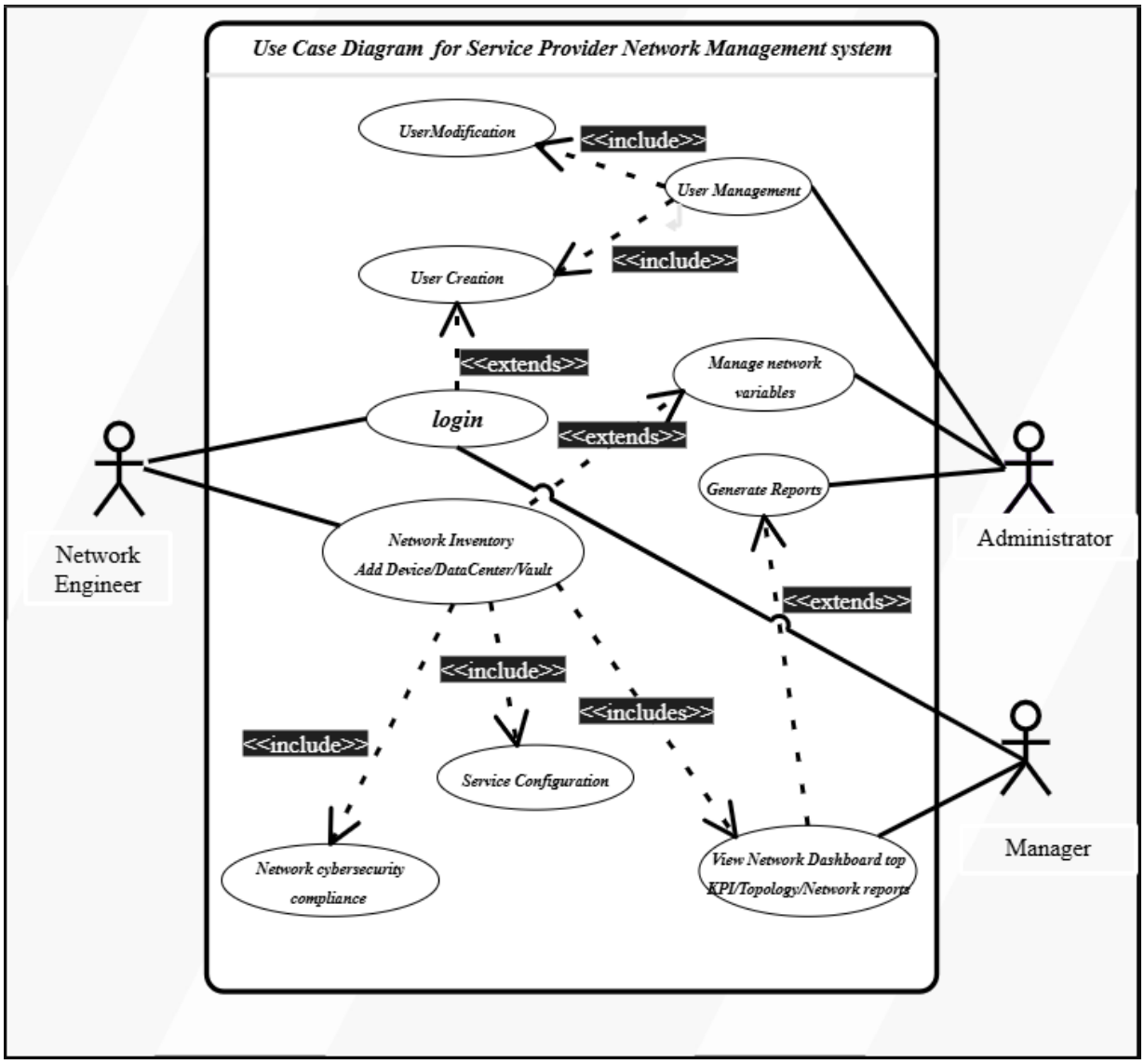


Figure 5.3:Use Case Diagram

5.9 Use Case Scenarios

These are tables showing the actors and descriptions of the major use cases that have been adopted into the project solution. Tables 5.1 to 5.6 summarize key use cases of the developed system. Table 5.1 covers user account management, while Table 5.2 outlines the process for user

deletion. Table 5.3 details viewing the network dashboard, and Table 5.4 focuses on executing network operations. Table 5.5 describes service deletion, and Table 5.6 addresses viewing system or service logs. Collectively, these use cases define the system's core functionalities and user interactions.

Table 5.1:Use Case description user account management

Use Case ID	001
Use Case Name	Create User
Actor	Network System Administrator
Description	Creates a user in the System
Pre-Conditions	User does not exist in the database
Post Conditions	User should be able to access app services
Major Inputs	Major inputs Username and Password
Major Outputs	Login Success
Includes	Access rights definition
Special Requirements	Employee Verifications
Assumptions	None

Table 5.2:Use case description for delete user

Use Case ID	002
Use Case Name	Delete user

Use Case ID	002
Actor	Network System Administrator
Description	Remove user from the system
Pre-Conditions	User is created in the system
Post Conditions	User is removed from the system
Major Inputs	Username
Major Outputs	User deleted
Includes	Remove all rights
Special Requirements	
Assumptions	User no longer needs access to system

Table 5.3:Use Case to view Network Dashboard

Use Case ID	003
Use Case Name	View Network Dashboard
Actor	Network Administrator/Network engineer
Description	View Network Dashboard
Pre-conditions	Successful authentication
Post Conditions	Logout from the system
Major inputs	Username & password

Use Case ID	003
Major Outputs	Network Dashboard with multiple KPI’s
Includes	Network Inventory
Special Requirements	Engineer verification
Assumptions	Network engineer is authorized to work with the system

Table 5.4:Use case description to perform network operation

Use Case ID	004
Use Case Name	Create Service(L2VPN) or L3VPN
Actor	Network Engineer
Description	Create service new customer service
Pre-conditions	Service is a l2vpn or L3vpn service
Post Conditions	Clear service definition and requirement
Major inputs	Service type
Major Outputs	Service successfully created
Includes	Devices exists on the system inventory
Special Requirements	Network Engineer verification
Assumptions	None

Table 5.5:Use case description to delete service

Use Case ID	005
Use Case Name	Delete service
Actor	Network Engineer
Description	Remove service prom the service
Pre-conditions	Service created in the system
Post Conditions	Service is removed from the system
Major inputs	Service name
Major Outputs	Service deleted
Includes	devices exist on the inventory
Special Requirements	Service requirement verification
Assumptions	Service owners is

Table 5.6:Use case description for view logs (system or service logs)

Use Case ID	006
Use Case Name	View logs
Actor	Network Engineer/Network administrator
Description	View system, inventory or services logs

Use Case ID	006
Pre-conditions	Management credentials
Post Conditions	View all logs
Major inputs	Service Name
Major Outputs	Customer logs
Includes	Network element information
Special Requirements	Network administrator verification
Assumptions	None

5.10 Sequence Diagram

Figure 5.4 illustrates the sequence of events and the operation of the service provider network automation system from the time a user logs in to when the system performs automated tasks such as managing network inventory, provisioning services, and ensuring cybersecurity compliance. It demonstrates how users, such as network engineers, administrators, and managers, authenticate themselves, access the network dashboard for insights, and initiate specific processes. For instance, network engineers can add or delete devices in the inventory or perform compliance checks, while managers can view network status. The figure also highlights how data flows through the system modules—authentication, inventory management, cybersecurity compliance service provisioning and dashboards. Ensuring seamless automation and operational efficiency.

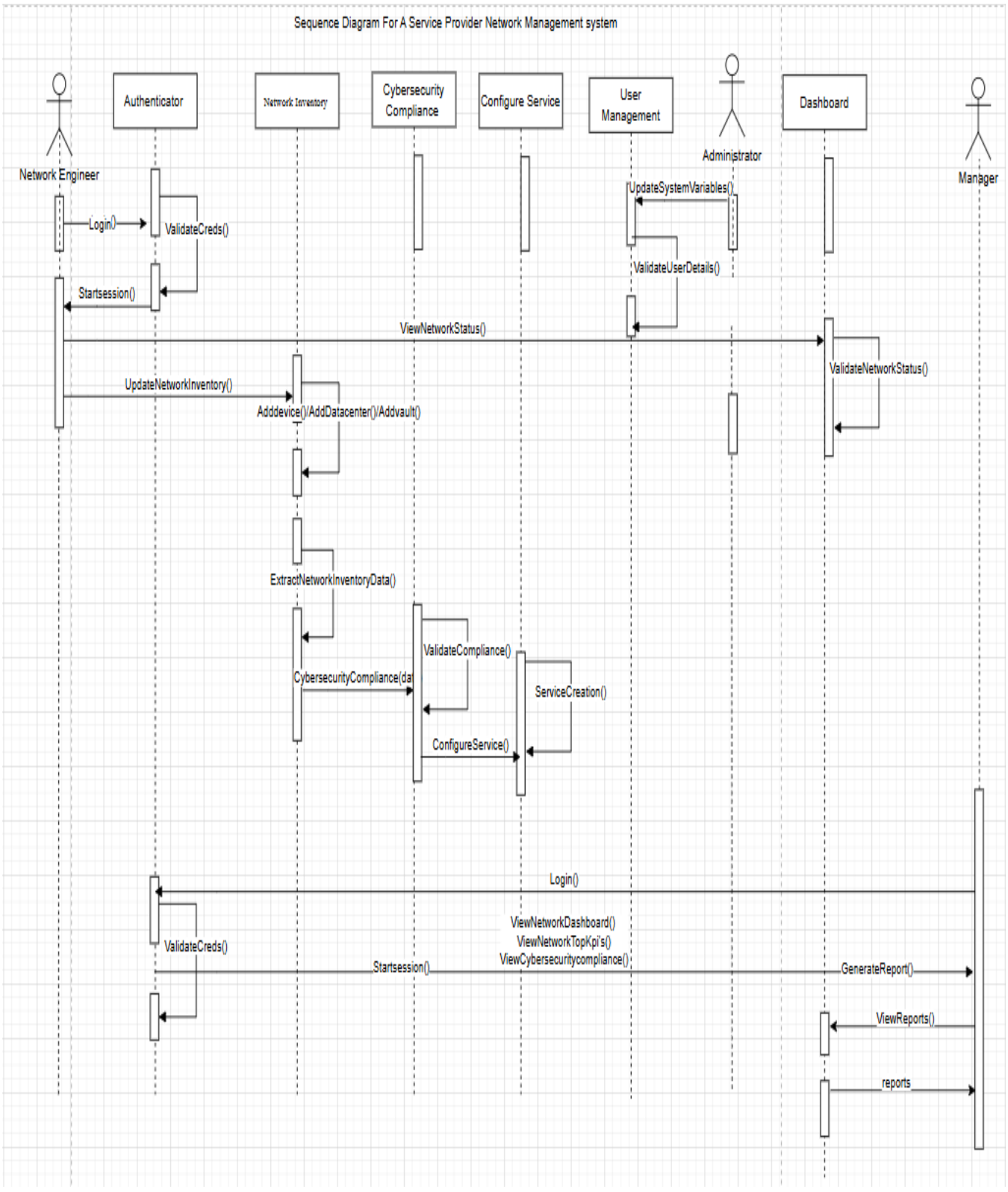


Figure 5.4:Sequence Diagram

5.11 Entity Relationship Diagram (ERD)

Figure 4.3 illustrates how various entities within the service provider network automation system interact with each other. It highlights the relationships between different database tables and their attributes, such as user authentication, network inventory management and cybersecurity compliance. The diagram showcases how data flows between entities, detailing attributes (columns) like compliance status, and network inventory. It demonstrates how these elements are interconnected, ensuring efficient data management and seamless operations within the system.

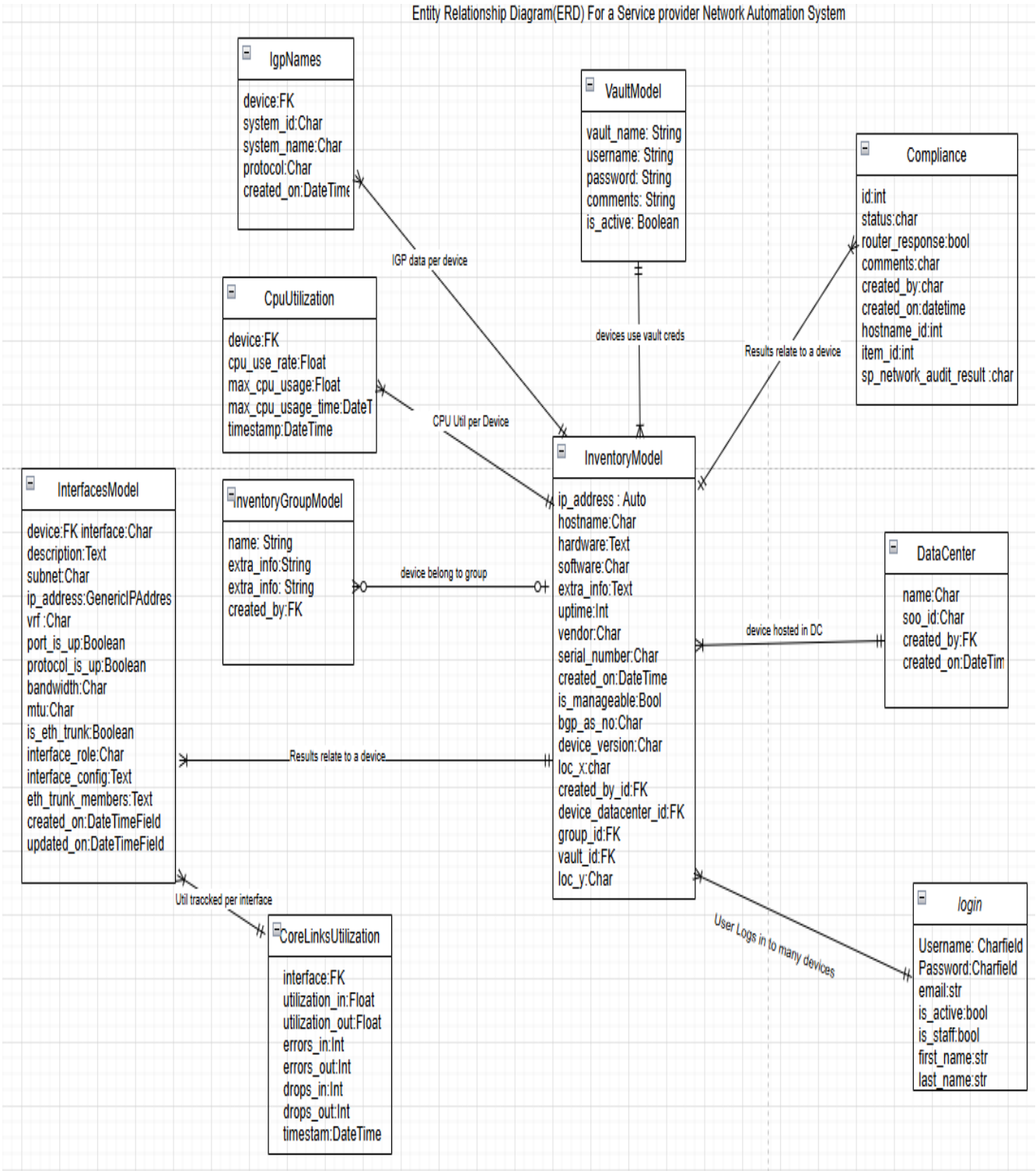


Figure 5.5:Entity Relationship Diagram

6: System Implementation and Testing

6.1 Introduction

This application was developed and tested on a service provider network lab infrastructure hosting Huawei routing equipment. Further, its breaks down the development and tests carried out into the two major parts of the client-side implementation, which would refer to the web application and the server-side implementation.

Also, it goes on to display how the administration portal was designed and implemented to ensure efficient support is put in place to run application. Implementation details reflect the technical environment of Safaricom Keya Limited. Organizations adopting this system should assess compatibility with their own infrastructure.

6.2 Implementation Environment

Django is a powerful web framework for Python that has gained widespread popularity since its release in 2005. Developed by Adrian Holovaty and Simon Willison, Django follows the model-template-view (MTV) architectural pattern, which is like the model-view-controller (MVC) pattern used in other frameworks. Django's philosophy emphasizes the principle of "Don't Repeat Yourself" (DRY), encouraging code reusability and maintainability. It provides a comprehensive set of tools and features that enable rapid development of secure and scalable web applications (S. Chen et al., 2020).

6.2.1 Some key components of Django include

i. Models: Django's Object-Relational Mapping (ORM) system allows developers to define database structures using Python classes. This abstraction layer simplifies database interactions and supports multiple database backends.

ii. Templates: Django's template language enables the separation of design from logic. It provides a powerful syntax for rendering dynamic content while maintaining clean HTML structure.

iii. Views: These are Python functions or classes that process requests and return responses. Views handle the logic of the application, interacting with models and rendering templates.

iv. URLs: Django's URL dispatcher maps URLs to views, allowing for clean and SEO-friendly URLs without relying on file extensions.

v. Forms: Django provides a robust form framework for handling user input, including validation, rendering, and processing of form data.

vi. Admin interface: One of Django's most praised features is its automatic admin interface, which provides a ready-to-use interface for managing application data.

vii. Authentication: Django includes a full-featured authentication system that handles user accounts, groups, permissions, and cookie-based user sessions.

viii. Middleware: These are hooks into Django's request and response processing, allowing for global modifications to Django's input or output.

ix. Django Vuexy Bootstrap: Offers Layout options for building next-generation web applications that follow the latest trends in web design and user experience.

Django's modular architecture and extensive ecosystem of third-party packages make it suitable for a wide range of web applications, from simple websites to complex, data-driven web services. Its emphasis on cybersecurity, scalability, and rapid development has made it a popular choice among developers and organizations of all sizes (Zignuts, 2025). Figure 6.1 depicts the interaction flow among Django modules, views, and templates.

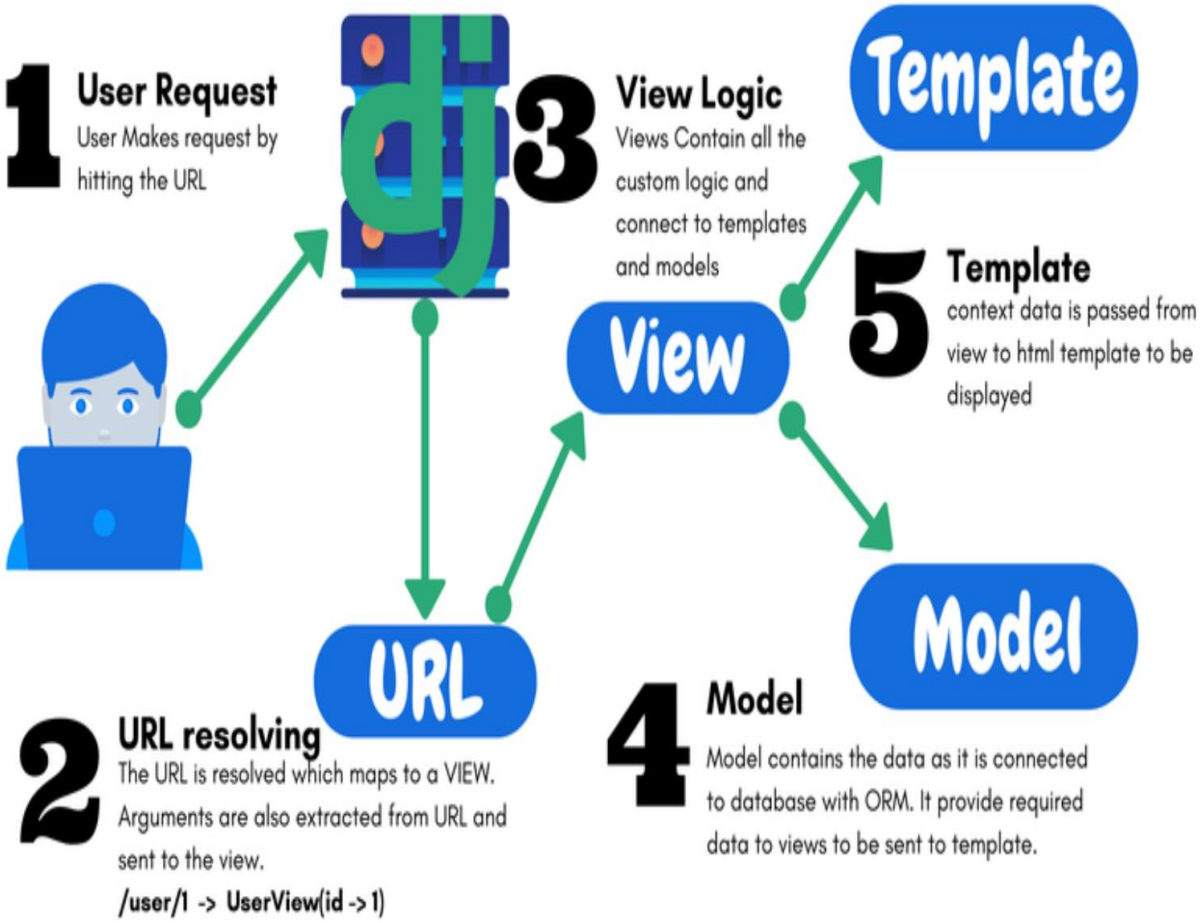


Figure 6.1:Django Module view template interactions (Zignuts, 2025)

6.3 Programming Language Used

Python, in conjunction with the Django web framework, has become a pivotal toolset for network automation, offering a robust and adaptable solution for managing intricate network infrastructures. Python's clarity, readability, and extensive library ecosystem make it an ideal choice for automating repetitive network tasks, enhancing efficiency, and minimizing human errors. The language's versatility enables the creation of scripts that can handle a wide range of network automation tasks, from configuration management to monitoring, inventory management, and cybersecurity implementation (Network Journey, 2024).

The synergy between Python and Django is particularly effective for network automation due to several key factors. Python's libraries, such as Netmiko and Paramiko, provide high-level interfaces for interacting with network devices, simplifying vendor-specific interactions. Django's Object-Relational Mapping (ORM) facilitates efficient database interactions, enabling

the creation of comprehensive network inventory, network compliance inventory and configuration management databases. This capability is crucial for maintaining an accurate view of the network infrastructure and tracking changes over time. Additionally, Django's templating system and form handling capabilities make it easy to build user-friendly web interfaces for network management tasks, promoting the adoption of automation tools within organizations and empowering staff to perform routine tasks safely. The extensive documentation, active community support, and continuous development of Python and Django ensure that network automation solutions built with these tools can evolve alongside changing network technologies and requirements (Opere, 2023).

6.4 Integrated Development Environment (IDE) used for the application development

PyCharm, a powerful Integrated Development Environment (IDE) specifically designed for Python development, offers a comprehensive suite of tools and features that make web application development efficient and user-friendly. At its core, PyCharm provides excellent support for popular web frameworks like Django, offering project templates that allow developers to quickly set up a project with the necessary directory structure and basic files. The IDE's intelligent code assistance is a standout feature, providing enhanced code comprehension and readability through syntax highlighting, error detection, and context-aware code completion specific to web development frameworks. This is complemented by PyCharm's robust debugging tools, which allow developers to step through their web application code, set breakpoints, and inspect variables in real-time, significantly streamlining the debugging process. Furthermore, PyCharm seamlessly integrates with version control systems like Git, making it easy for developers to manage code changes and collaborate with team members effectively (Kumar & Nandal, 2024b).

The IDE's capabilities extend beyond basic coding and debugging. PyCharm offers sophisticated database tools for managing connections, writing and executing queries, and visualizing data, which is crucial for web applications that require database interactions. PyCharm's run configurations feature allows developers to easily set up and manage different environments for their web applications, facilitating quick testing and deployment. Additionally, the IDE boasts an extensive plugin ecosystem, enabling developers to extend its functionality to suit specific web development needs. These comprehensive features, combined with PyCharm's

intuitive interface and regular updates, make it a go-to choice for Python web developers, enabling them to create sophisticated, efficient, and scalable web applications with ease (Ramakrishnan, 2022).

Figure 6.2 below show an example of PyCharm professional environment Django project creation and initialization.

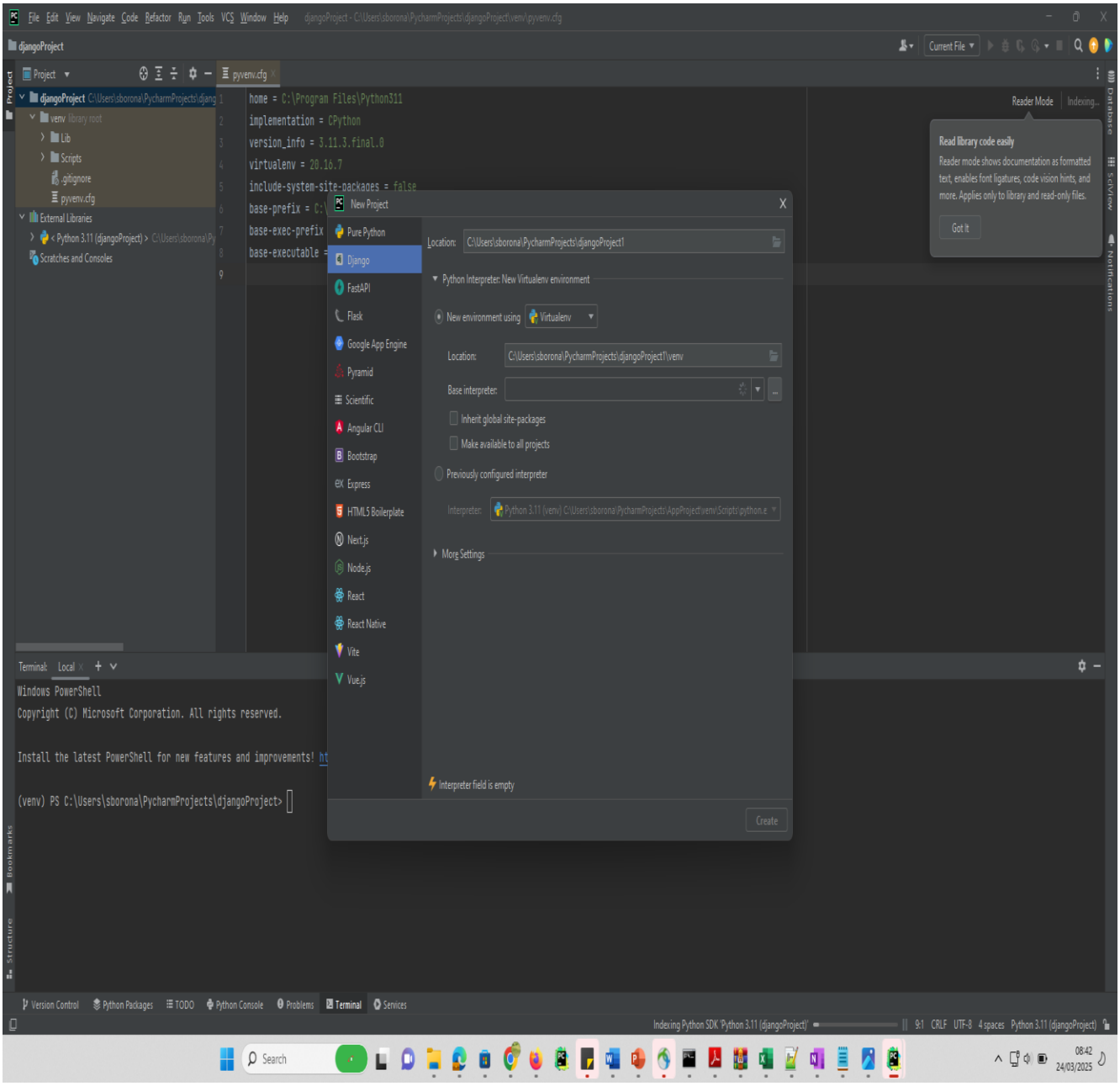


Figure 6.2:Pycharm Professional View

6.5 The Network Management Web Application Implementation

This network management application web Portal is fully developed and tested on actual network devices. The application file structure is presented in the Figure 6.3 below. This is how the application appears on the PyCharm professional development platform. The application structure is broken down into major noticeable parts. We have six applications notably admin, compliance, dashboards, devices, digital map, provisioning, each application folders consisting of main python files for example devices application comprise of models.py, tasks.py, url.py, views.py, admin.py, _init_.py, apps.py, forms.py and template folders which holds the html files. The major python files are urls.py, tasks.py, views.py, models.py. Further, the entire application also hosts global files like manage.py to support execute the entire application, static folder to host needed files like images, and settings file holds of entire project settings.

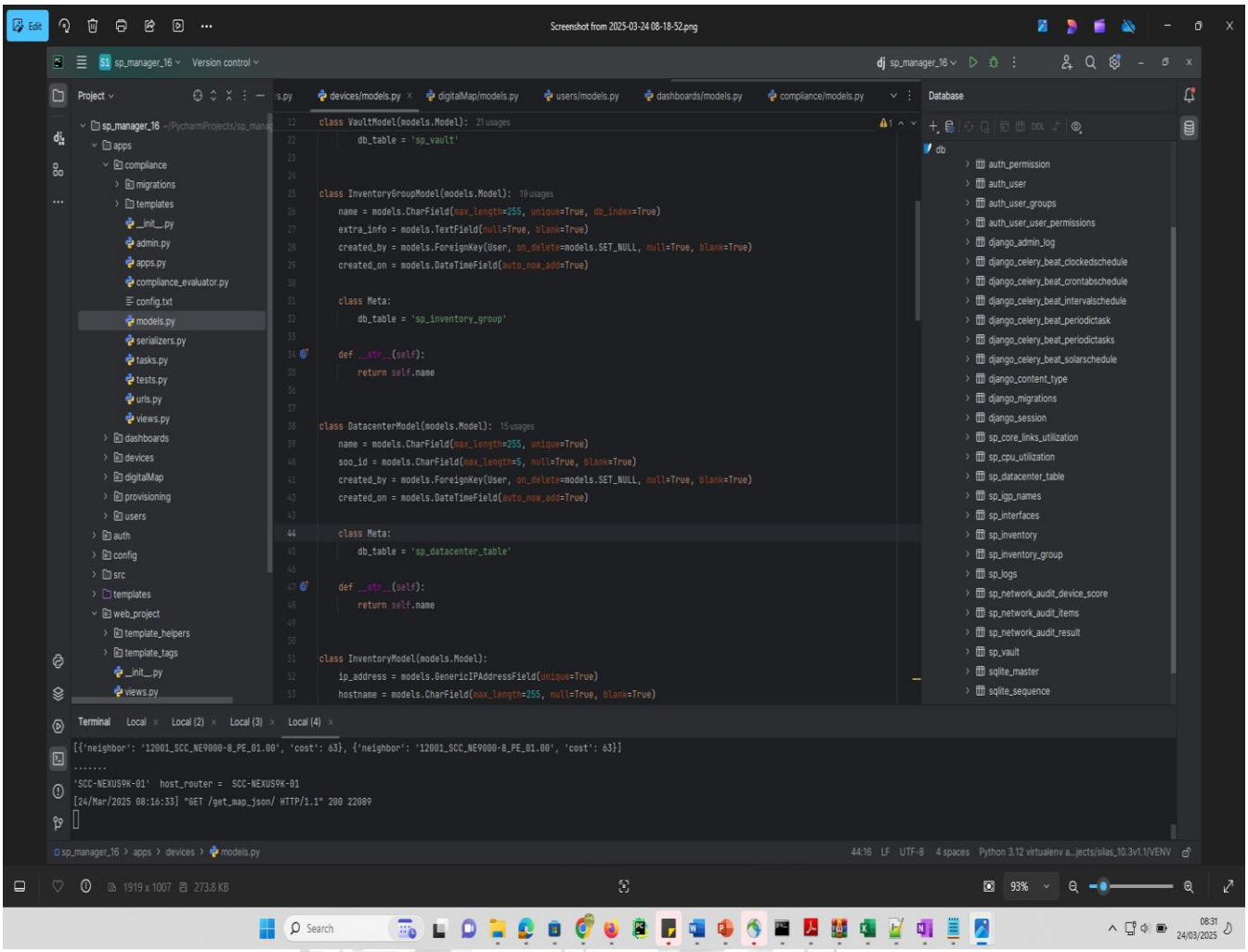


Figure 6.3:Application Structure on PyCharm Professional

6.5.1 Application Authentication

This homepage allows validated users to log into the system by providing a previously created username and password. It further implements cybersecurity features by denying access to the system if the username or password is not correct. The password provided should match the user‘s name as defined in the database system.

As shown figure 6.4 below, the authentication is responsible for application access where users login to access various other applications listed above.

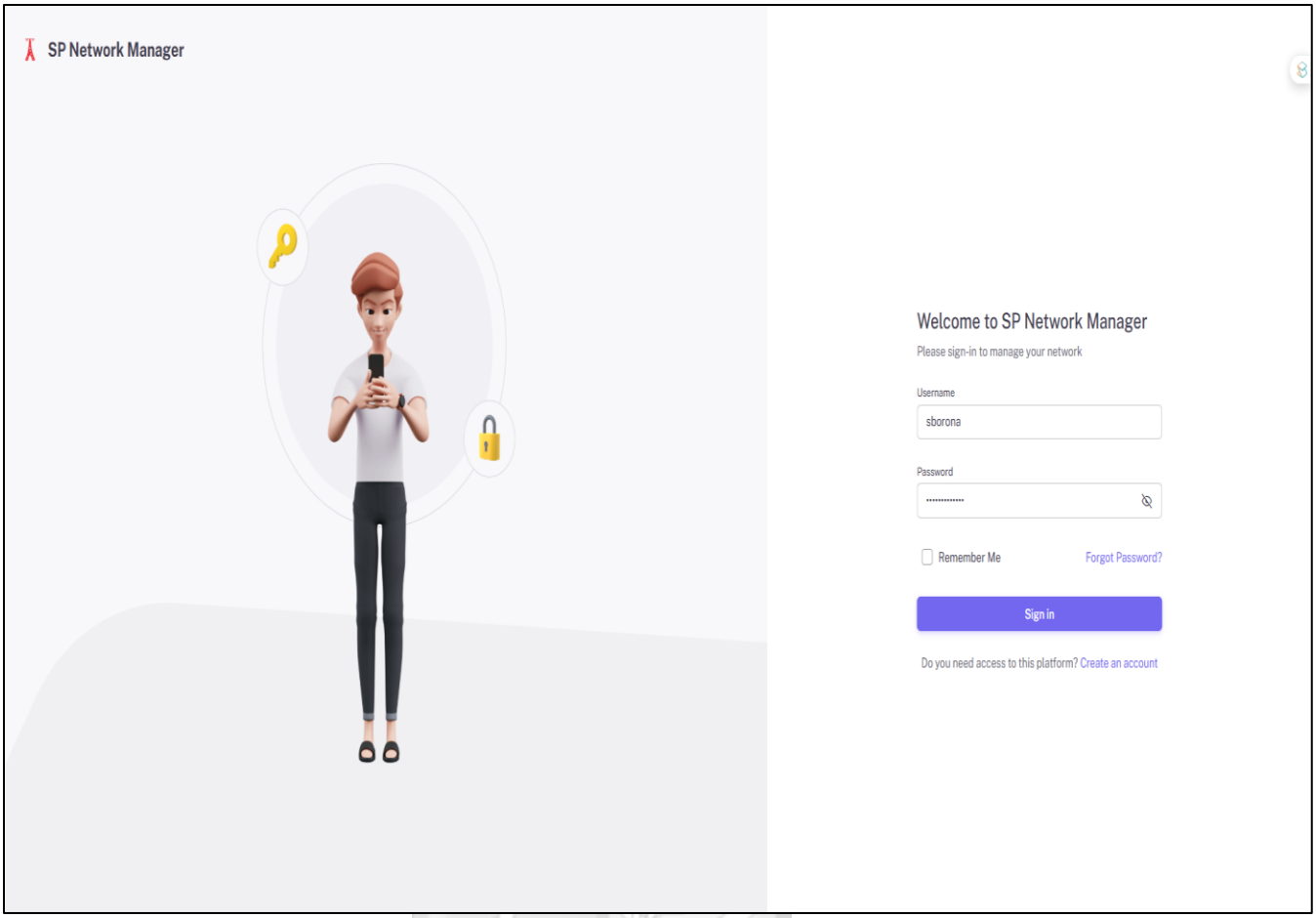


Figure 6.4:Authentication App

6.5.2 Network Dashboard Application

Upon successful authentication, users are seamlessly directed to the contact page, which serves as the network dashboard application (as illustrated in Figure 6.5). This comprehensive dashboard provides a single-pane-of-glass view, offering real-time insights into the network's performance and configuration. The interface presents critical network information, including network interconnectivity, link utilization, link cost, and bandwidth, allowing users to quickly assess the overall health of their network infrastructure. To enhance user experience and productivity, the application offers customization features, enabling users to modify the layout of network elements according to their preferences and priorities. This intuitive and flexible dashboard

empowers network administrators efficiently monitor, analyze, and manage their network infrastructure from a centralized interface.

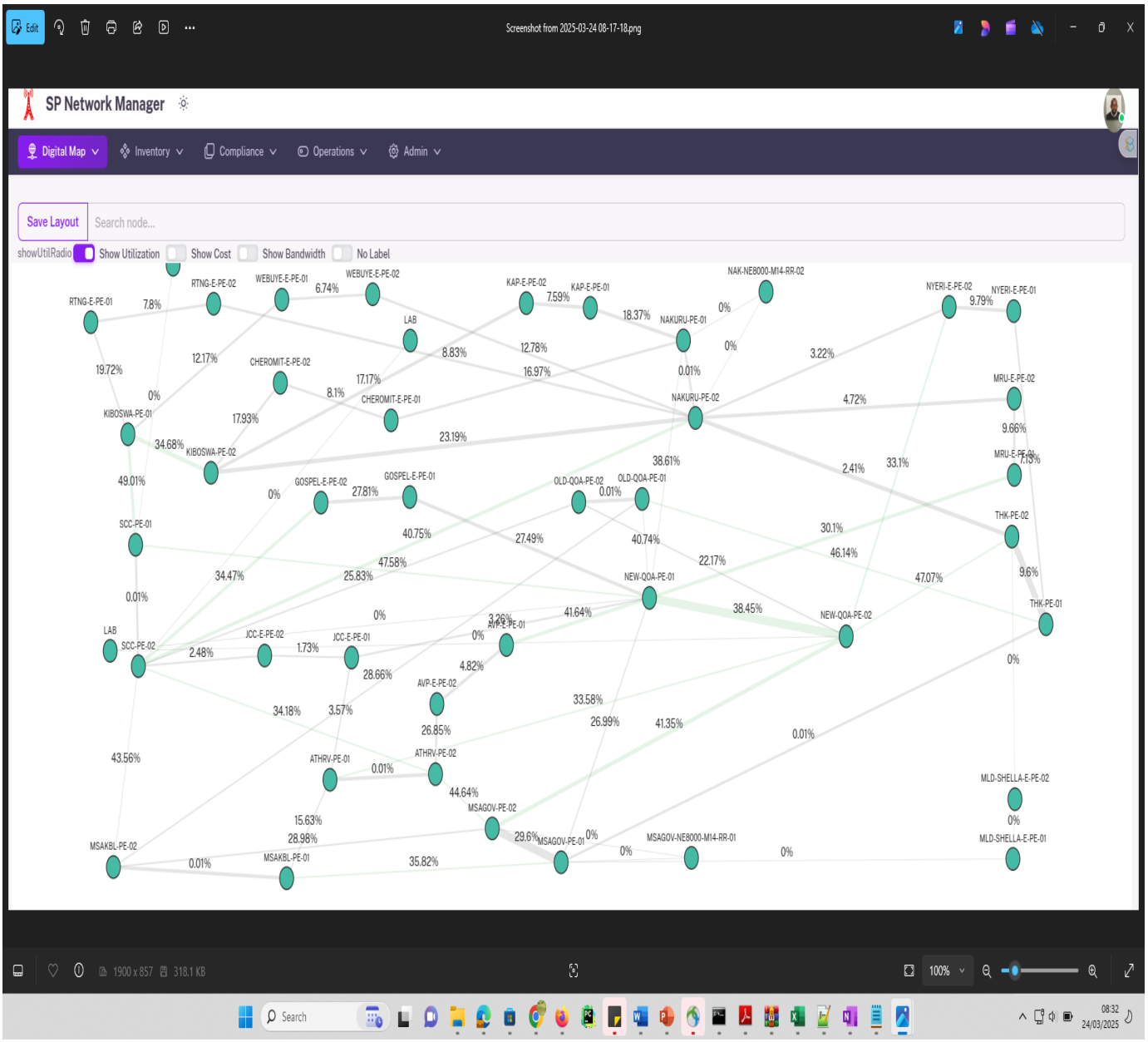


Figure 6.5:Dashboard Application

6.5.3 Network Inventory application

Figure 6.6 showcases the network inventory application; a powerful tool designed to streamline network management and provide comprehensive visibility into network assets. The application features a user-friendly interface with a versatile drop-down menu, offering various

functionalities to enhance network inventory management. Users can effortlessly list devices across the network, add new data centers, and incorporate additional devices into the application, all from a centralized location. The dashboard serves as a quick reference point, displaying critical metrics such as the total number of managed devices and available data centers, enabling administrators to gain instant insights into their network infrastructure. This intuitive design allows for efficient asset tracking, simplified device management, and improved operational oversight, making it an invaluable resource for network administrators seeking to maintain an up-to-date and accurate inventory of their network resources.

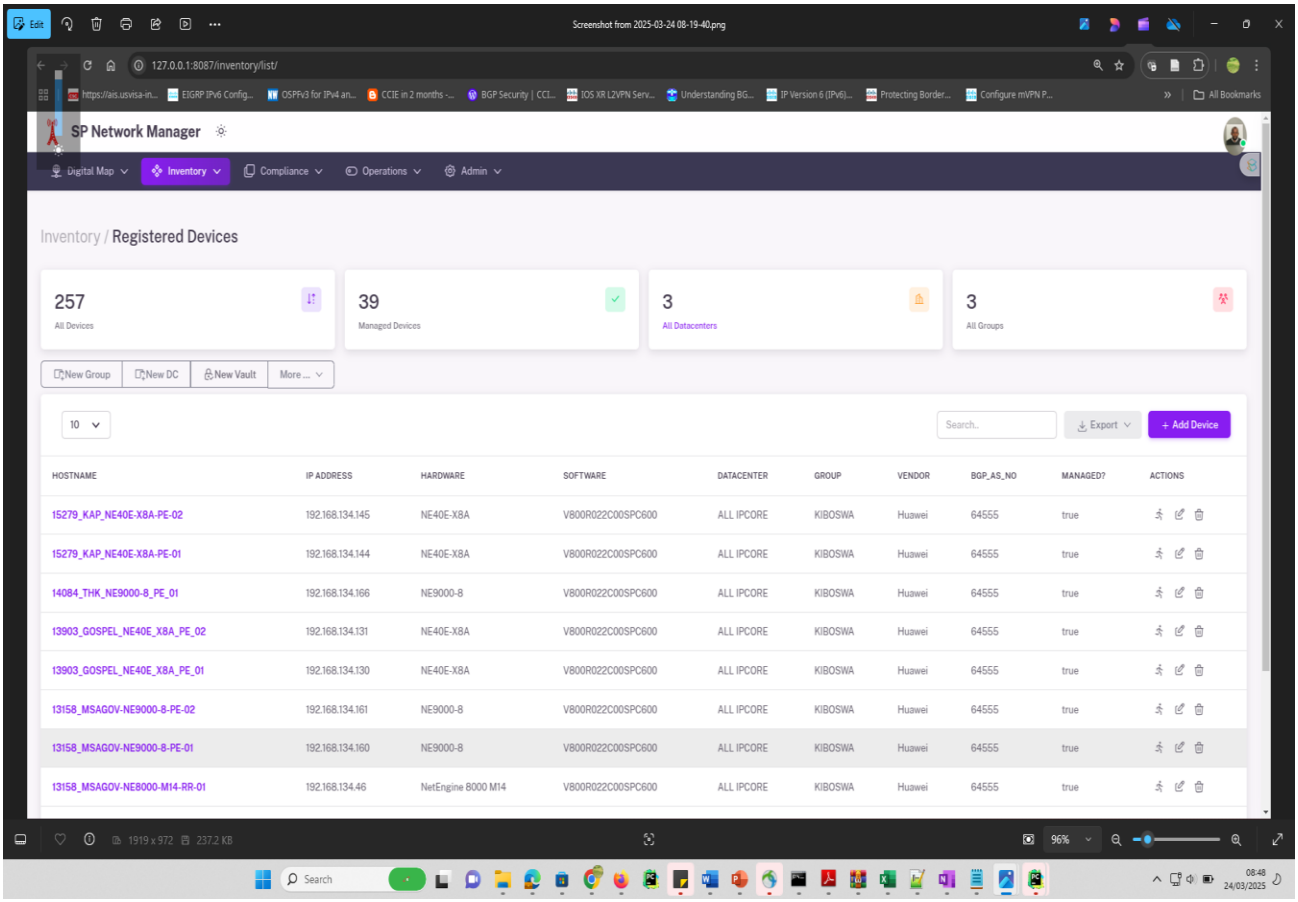


Figure 6.6:Network Inventory Application

6.5.4 Network Cybersecurity Compliance application

Figure 6.7 illustrates the network cybersecurity compliance application, a crucial tool for maintaining network cybersecurity and regulatory adherence. This comprehensive interface

presents both instance compliance reports and network reports, providing a holistic view of the network's cybersecurity compliance status. The cybersecurity network compliance report section offers detailed insights, including group compliance reports that categorize items based on their compliance status. Users can quickly identify items that have passed cybersecurity compliance checks, those that have failed, and any that have been granted cybersecurity exemptions. This granular breakdown enables network administrators to prioritize cybersecurity remediation efforts, address non-compliant elements promptly, and maintain an accurate record of the network's overall cybersecurity compliance posture.

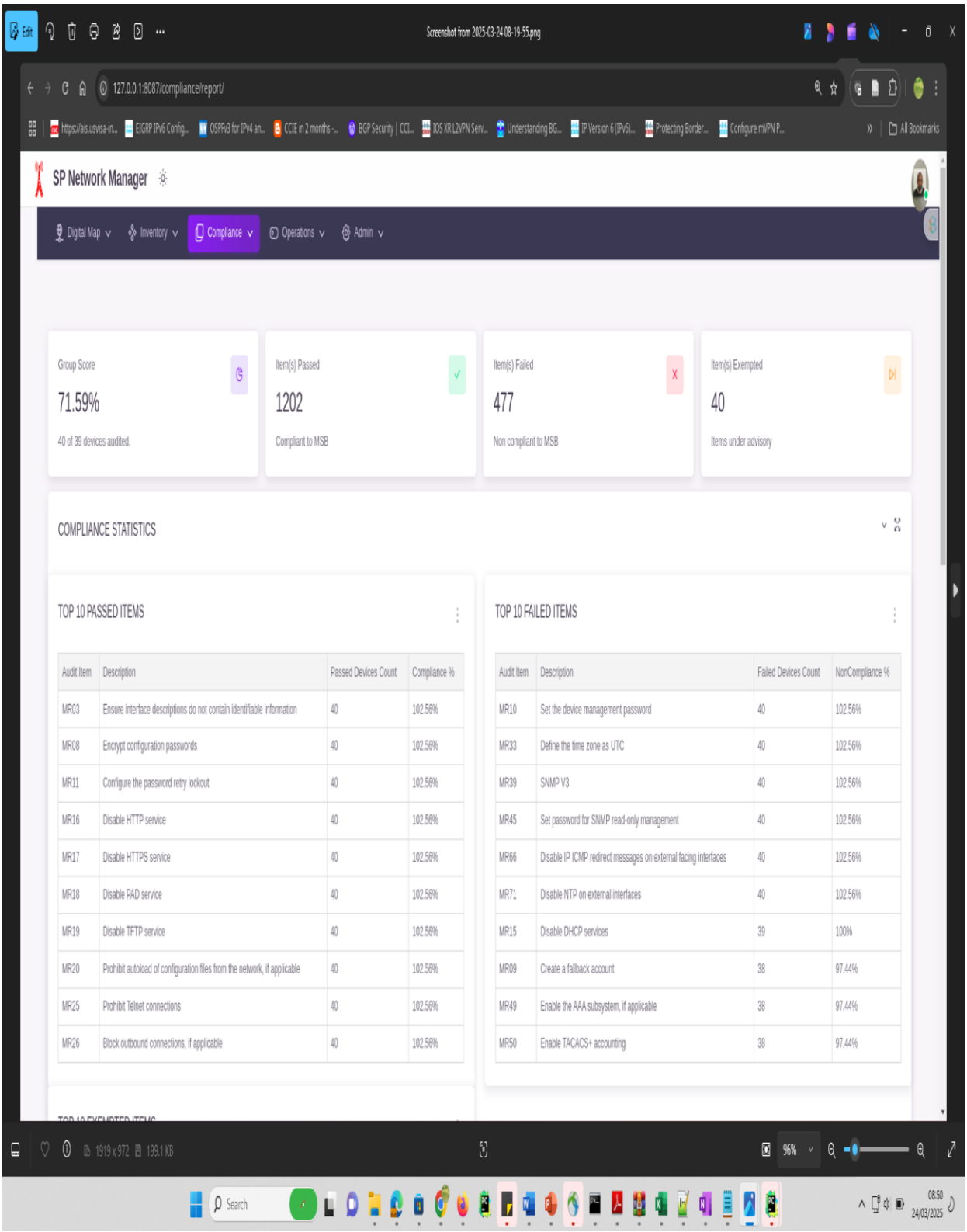


Figure 6.7:Network Compliance Report

6.5.5 Network Operation application

Figure 6.8 showcases a service provisioning instance within the application, designed to enhance network configuration management and reduce errors. This feature allows engineers to push configurations directly to the network devices, streamlining the provisioning process and minimizing the risk of manual generated network configuration script errors. By automating network provisioning, the application enables more efficient and accurate configuration changes across multiple devices simultaneously. This approach not only saves time but also ensures consistency in network configurations, improving overall network stability and cybersecurity. The service provisioning instance incorporates version control and validation checks, further safeguarding against potential misconfigurations that could lead to network disruptions.

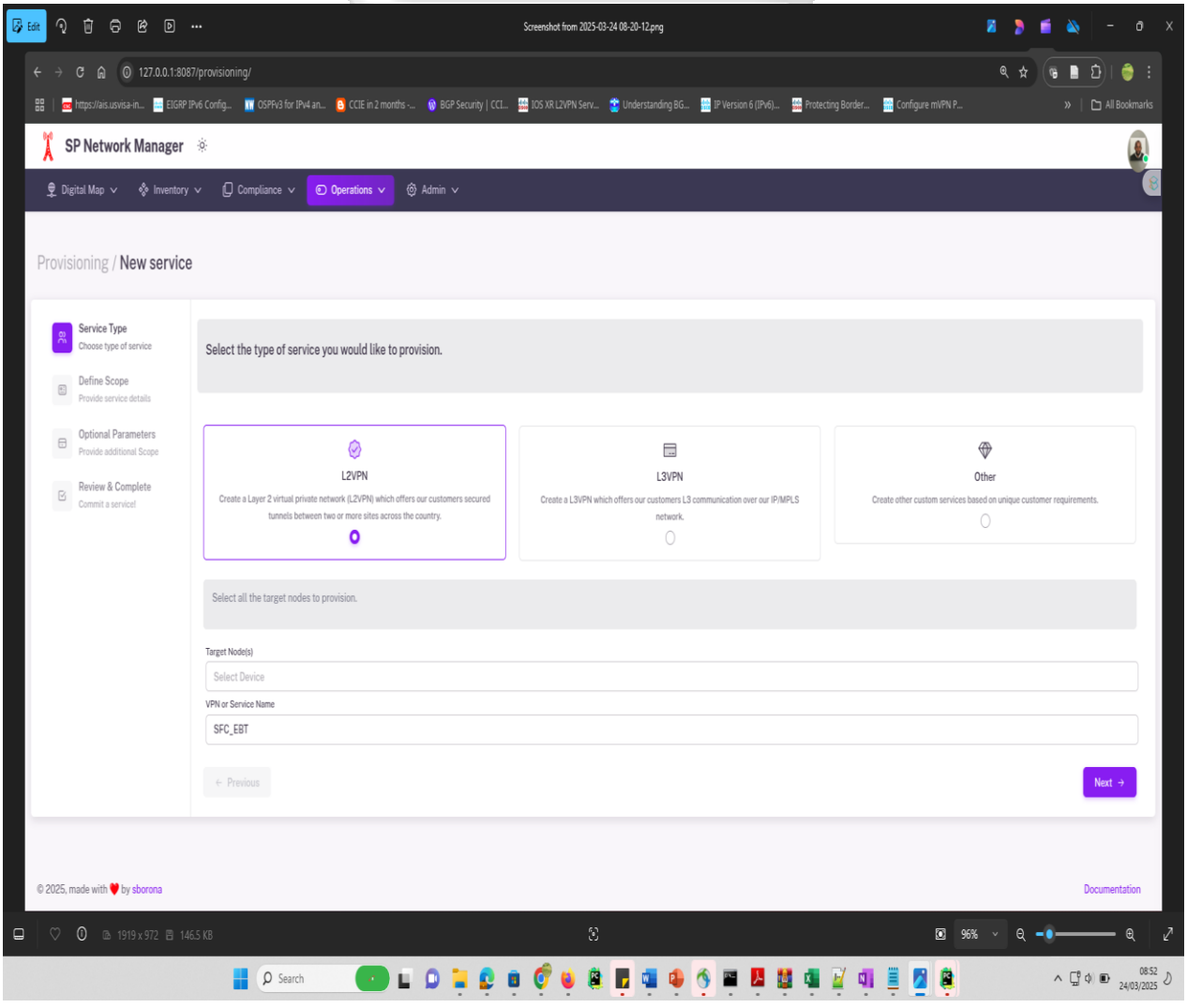


Figure 6.8:Network Operations application

6.5.6 The Administration module

Figure 6.9 illustrates the system administration interface, a crucial component for managing user access and permissions within the network management platform. This interface empowers the system administrator with essential user management capabilities. From this centralized console, administrators can efficiently perform various tasks related to user account management. They can create new user accounts, granting access to appropriate individuals as the organization grows or roles change. Similarly, the interface allows for the deletion of user accounts when necessary, such as when employees leave the organization, or their roles no longer require system access. Additionally, while not explicitly mentioned, it is likely that this interface also provides options for modifying existing user accounts, resetting passwords, and assigning or adjusting user permissions. This comprehensive set of tools ensures that the system administrator can maintain tight control over who has access to the network management system, thereby enhancing overall cybersecurity and operational efficiency.

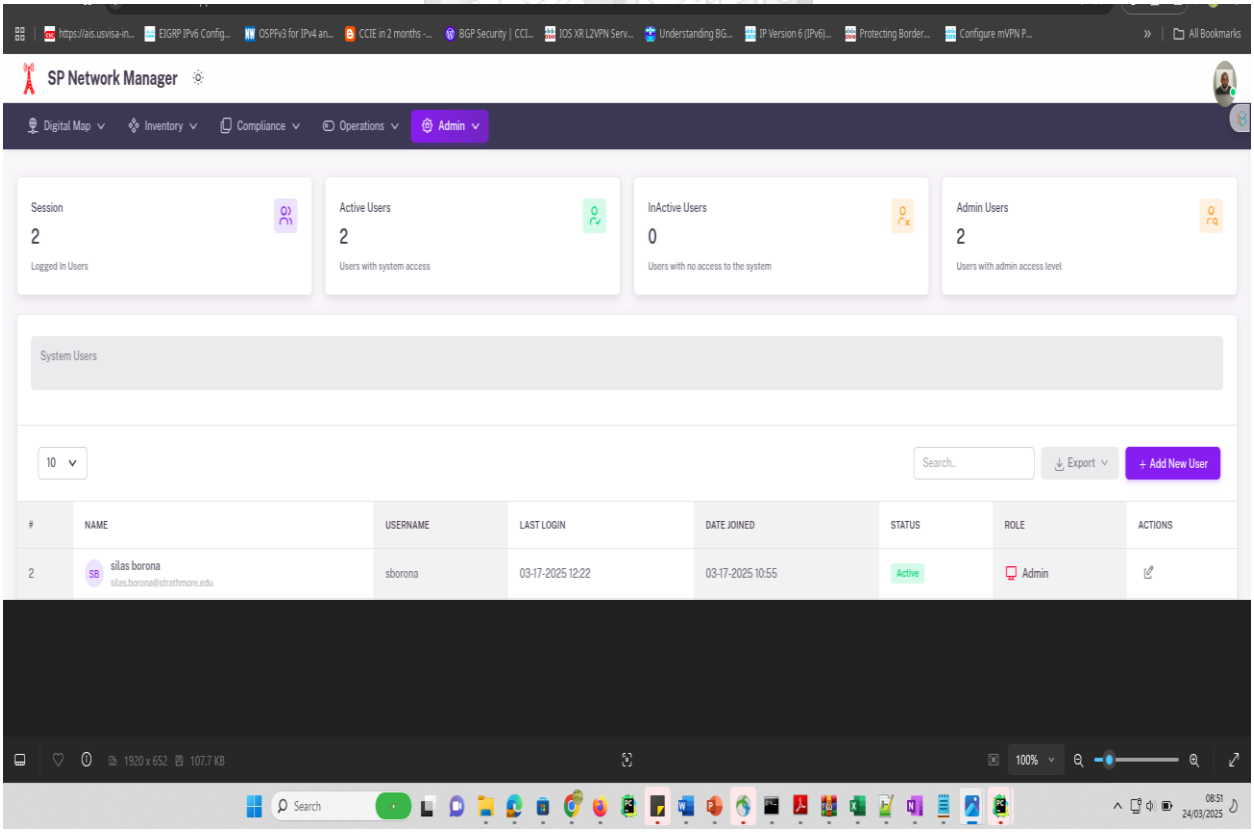


Figure 6.9:Application administration

6.6 System Testing

6.6.1 Testing Paradigm

During the development of the Django and python-based service provider network management application, both unit and integration testing were carried out.

6.6.2 Testing Results

Table 6.1 provides an overview of the authentication module testing.

Table 6.1:Authentication Module

Test	Test data	Expected result	Result	Pass/Fail
Registration	silas.borona@strathmore.edu	User is successfully registered and is taken to the login screen.	As expected	Pass
Successful login after registration	Registered username, email address and password	Successful login if user is verified	As expected	Pass
Log in as different user	Email address registered under the specific role: Network Engineer, Network Manager and Administrator	Tale user to the correct dashboard after verification	As expected	Pass

Network Engineer Module

These are some of the actions a network engineer can perform within the application after logging in. Table 6.2 provides an overview of the network engineer module tests and its associated functions.

Table 6.2:Network Engineer Module

Test		Expected result	Result	Pass/Fail
Network Engineer successfully accessing the homepage	Registered network engineer username, password Email Address	User is successfully logged in and is taken to the homepage	As expected	Pass
Add device	Add network device to the Inventory, add datacenter and vault information	Network engineer successfully added a device/data center/vault to the inventory.	As expected	Pass
Device Cybersecurity Compliance	Network engineer to run network device cybersecurity compliance	Cybersecurity compliance completed	As expected	Pass
Configure service	Configuration of L2vpn or L3vpn service	Cybersecurity compliance completed	As expected	Pass
View Dashboard	Network status	View network topology, KPI's, cybersecurity compliance	As expected	Pass

Table 6.3 provides an overview of the network manager tests and its core functions.

Table 6.3:Network Manager Module

Test		Expected result	Result	Pass/Fail
Network Manager successfully	Registered network manager Email Address	User is successfully logged in and	As expected	Pass

accessing the homepage		is taken to the homepage		
View Dashboard	Network status details (Network topology status, Network KPI status)	Successful view of network details	As expected	Pass
View Network reports	Network details (device details, compliance reports)	Summary of Network reports from inventory and compliance is successfully shown.	As expected	Pass

Table 6.4 provides an overview of the system administrator tests.

Table 6.4:System Administrator Module

Test		Expected result	Result	Pass/Fail
Administrator successfully accessing the homepage	Registered Administrator Email Address	User is successfully logged in and is taken to the homepage	As expected	Pass
Add User	User details	Successful adding of user details	As expected	Pass
Update User	New user details	Successful update of user details	As expected	Pass
View network reports	Network details	Summary of network reports from drivers is successfully shown.	As expected	Pass
Update network inventory information	New network inventory information (data center, device, vault)	Successful updating of the new network inventory information	As expected	Pass

7: Discussion and Findings

7.1 Introduction

The telecommunications industry is undergoing rapid technological transformation, necessitating the adoption of efficient and precise network management solutions. This dissertation presents the design and development of a web-based application utilizing the Django framework and Python programming language, aimed at addressing the practical challenges encountered by network engineers in their routine operations. The research seeks to mitigate the limitations associated with manual network management and the time-consuming nature of traditional network functions by enhancing the efficiency, accuracy, and responsiveness of network visualization, configuration, and maintenance processes. By leveraging the robustness of Django and the flexibility of Python, this study introduces an innovative approach that incorporates synchronous network data handling, automated reporting capabilities, and adherence to comprehensive network cybersecurity compliance standards.

The scope of this research entails a comprehensive examination of contemporary network management practices, the identification of prevailing challenges, and the subsequent design and implementation of a web-based solution tailored to the unique requirements of the telecommunications industry. Through a multidisciplinary methodology encompassing an extensive literature review, field-based surveys, and the development of a practical application, this study seeks to advance the discourse on network management automation and cybersecurity compliance within the domain of telecommunications engineering. The findings and recommendations derived from this research have the potential to substantially enhance the operational efficiency and effectiveness of network engineers, thereby laying the groundwork for the adoption of more advanced, integrated, and intelligent network management frameworks in the telecommunications sector.

7.2 Summary of Findings

The research conducted for this master's dissertation revealed several critical findings that emphasize the importance of developing a Django-based web application for network management in the telecommunications field. Primarily, network engineers consistently face challenges in accessing up-to-date information about network infrastructure while performing day to day activities, leading to inefficiencies and potential errors in configuration and maintenance tasks. The study found that engineers spend a significant portion of their time

communicating with multiple personnel to retrieve or verify information, which could be more efficiently accessed through an application. Additionally, the research highlighted the inadequacy of current methods for capturing and updating network information.

Moreover, the analysis of existing network management applications in the market revealed a gap in solutions specifically tailored to the needs of telecommunications engineers. While some applications offer partial functionality, none provided a comprehensive solution that integrated real-time data access, cybersecurity update capabilities, and automated reporting features. The development and testing of the Django and python-based system demonstrated significant improvements in task completion times, data accuracy, and overall efficiency of engineer's operations. Engineers reported a high level of satisfaction with the application's user interface and functionality, particularly appreciating the ability to access and update network information in real-time. These findings strongly support the dissertation proposition that a well-designed, Django and python-based web application can significantly enhance the effectiveness of network management operations in the telecommunications industry.

7.3 Discussion

The findings of this study reinforce and extend the insights presented in the literature review regarding the challenges and opportunities in network management automation.

First, the persistent challenges of manual network management identified in this research such as inefficiency, high error rates, and poor scalability are consistent with the literature (Sarrigiannis et al., 2022; Wang et al., 2018). As highlighted in Section 2.3, manual approaches lead to prolonged fault resolution times and increased operational costs, a trend confirmed by the engineers surveyed in this study who reported significant time lost to information retrieval, provisioning processes, cybersecurity compliance and configuration errors. This direct alignment underscores the real world impact of the issues described by (Ramesh et al., 2023).

Second, the effectiveness of automation frameworks, particularly those built with Python and Django, is corroborated by both the literature and this study's results. The literature review (Section 2.5) notes that tools such as Ansible, Chef, and Salt have improved efficiency and reduced human error, but also points out their limitations such as steep learning curves, lack of comprehensive solutions, and integration challenges (Redhat, 2024; Whizlabs, 2024). This study's Django-based solution addresses several of these gaps by offering a user-friendly interface, network complexity abstraction, real-time data access, and integrated cybersecurity

compliance, service provisioning, network topology dashboards thus providing a more holistic approach than many existing tools (as discussed in Section 2.7).

Moreover, the study's finding that automated dashboards improved compliance rates directly supports the literature's argument that automation enhances cybersecurity and operational reliability. However, while previous studies such as (Muhammad & Munir, 2023) noted limitations in user interface and cybersecurity hardening, this research demonstrates that leveraging Django's robust ORM and REST framework can overcome some of these barriers, improving both usability and security.

Unlike previous case studies that relied heavily on simulation environments (Muhammad & Munir, 2023), this study incorporated real-time feedback from practicing engineers, real network use cases, which led to higher reported satisfaction. This suggests that involving end users in the development and testing phases, as recommended in agile methodologies (Section 3.7), is crucial for successful automation projects.

Additionally, while the literature points out that many automation tools focus on isolated tasks and require significant human oversight (Section 2.7), the integrated approach adopted in this study combining inventory management, compliance, operations and reporting demonstrates the value of a unified platform. This is a significant advancement over the fragmented solutions discussed by (Muhammad & Munir, 2023).

The results have several implications for both practice and research. Practically, they suggest that organizations should prioritize the development of integrated, user-friendly automation platforms that address the full spectrum of network management needs, rather than relying on piecemeal solutions. Theoretically, the success of the Django-based system supports the Network Automation Theory (Section 2.1.1), which posits that strategic automation leads to more agile and resilient infrastructures.

However, the study also highlights ongoing challenges, such as the need for better support for multi-vendor environments and the limitations of simulation-based testing. Future research should focus on deploying such systems in live, heterogeneous networks and exploring the integration of AI-driven predictive analytics (as suggested in Section 2.5.3) to further enhance automation capabilities. The design is based on Safaricom Kenya Limited case study. Broader validation and adaptation are required before applying this architecture to other MNOs or regions.

7.3.1 Significance considering Research Questions

i. RQ1: The study confirms that manual network management is rife with inefficiency and error prone, as detailed in both the literature and empirical findings.
ii. RQ2: Existing frameworks have limitations, but a Django-based solution can overcome many of these, especially in usability, efficiency comprehensiveness and integration.
iii. RQ3: The design and development of a unified automation platform was feasible and effective, as demonstrated by improved efficiency and user satisfaction.
iv. RQ4: Testing in a real-world environment is essential, and the study's approach offers real network testing and results.

This research demonstrates a clear assessment of the network automation landscape, critically evaluating both the strengths and limitations of current tools and frameworks. By situating the findings within the context of existing literature, the discussion not only validates previous work but also provides new insights and directions for the field.

8: Conclusion

8.1 Introduction

This dissertation explores the development of a Django-based web application for network management in the telecommunications industry. The research addresses the challenges faced by network engineers while managing networks. By leveraging Django's robust framework and Python's versatility, this study proposed and developed an innovative solution that integrates real-time network data access, immediate network insights, automated reporting, network operations and cybersecurity management for telecommunications network engineers.

The scope of this research included a comprehensive analysis of current network engineering practices, identification of key challenges, and the design and implementation of a web-based network management system tailored to the telecommunications industry's specific needs. Through literature review, surveys, and practical application development, this dissertation contributes to the body of knowledge in network management automation and network computing within the telecommunications service provider engineering context.

This research is constrained by the academic context in which it was conducted, relying primarily on survey data collected from a single organization. As a postgraduate student, the researcher did not have the access or role of a consultant within the firm, which limited the ability to perform comprehensive technical audits or direct observations of network processes. These external constraints affect the generalizability and depth of the findings, which should be interpreted with caution. This limitation is acknowledged throughout the dissertation, including in the methodology, results, design, and discussion chapters, to ensure transparency and contextual understanding of the study's scope.

8.2 Conclusion

The Django-based network management system developed in this dissertation represents a significant advancement in addressing the challenges faced by service providers network engineers. The research findings demonstrate that this solution effectively improves tasks completion, cybersecurity compliance, efficient network management, service provisioning reliable network resources inventory, and automated reporting, leading to enhanced efficiency and accuracy in network management.

The implementation of this system has shown promising results in streamlining the workflow of network engineers, reducing time spent on network configuration, reporting,

cybersecurity compliance and manual day to day network inventory and real-time visualizations of network key performance indicators (KPI's). The application's ability to function adequately, coupled with its user-friendly interface, addresses critical needs identified in the research. However, there are areas for future improvement and expansion, including device and platform compatibility, extend application service offering to the service provider access network, enhanced cybersecurity measures, and standardization across different service providers and routing vendors equipment.

8.3 Recommendations

Based on the research findings, several key recommendations have emerged. First, conducting a comprehensive study with a diverse pool of networks and systems is crucial to further validate the application's effectiveness in real-world scenarios. This feedback will be invaluable in refining the user interface and functionality. Second, expanding platform compatibility to include a phone application will enhance the system's versatility and ensure continuous access to critical information regardless of the available device.

Standardizing inputs and outputs across different service providers and vendor platforms is also recommended. This standardization will facilitate broader adoption of the system across various telecommunications companies. Additionally, implementing advanced cybersecurity measures, such as multi-factor authentication, support and end-to-end encryption, is essential to protect sensitive network information from cyber threats.

8.4 Future Work

Future development of the Django-based network management system should focus on achieving true device and platform independence. This includes optimizing the user interface for various screen sizes and orientations, as well as developing versions compatible with iOS and Windows mobile platforms. Such enhancements will significantly broaden the application's user base and utility across different organizational environments.

Further work should also strengthen on improving the system's performance and scalability to support high transaction volumes. This may involve optimizing database queries, implementing caching mechanisms, and exploring advanced Django features for high-performance applications. Continuous refinement of the user interface to meet international usability standards will be crucial in ensuring long-term adoption and user satisfaction. These improvements will position the Django and python-based network management system as a

cutting-edge solution for telecommunications field, driving efficiency and innovation in the sector.

References

Ahmed, S. K. (2024). How to choose a sampling technique and determine sample size for research: A simplified guide for researchers. *Oral Oncology Reports*, *12*, 100662. https://doi.org/10.1016/j.oor.2024.100662

Arinze, E. D. (2024). Enhancing Network Management Practices in East Africa: A Comprehensive Review. *IDOSR JOURNAL OF COMPUTER AND APPLIED SCIENCES*, *9*(2), 36–42. https://doi.org/10.59298/JCAS/2024/92.3642

Bellamkonda, S. (2023). Automating Network Security with Ansible: A Guide to Secure Network Automation. *International Journal Of Multidisciplinary Research In Science, Engineering and Technology*, *06*(09). https://doi.org/10.15680/IJMRSET.2023.0609021

Bisht, R. (2023, December 16). *What Are Sampling Methods? Techniques, Types, and Examples | Researcher Life*. https://researcher.life/blog/article/what-are-sampling-methods-techniques-types-and-examples/

Bolettieri, S., Bui, D. T., & Bruno, R. (2022). Towards end-to-end application slicing in Multi-access Edge Computing systems: Architecture discussion and proof-of-concept. *Future Generation Computer Systems*, *136*, 110–127. https://doi.org/10.1016/j.future.2022.05.027

Bringhenti, D., Marchetto, G., Sisto, R., & Valenza, F. (2024). Automation for Network Security Configuration: State of the Art and Research Trends. *ACM Computing Surveys*, *56*(3), 1–37. https://doi.org/10.1145/3616401

Brown, L., Marx, E., Bali, D., Amaro, E., Sur, D., Kissel, E., Monga, I., Katz-Bassett, E., Krishnamurthy, A., McCauley, J., Narechania, T., Panda, A., & Shenker, S. (2024). An

Architecture For Edge Networking Services. *Proceedings of the ACM SIGCOMM 2024 Conference*, 645–660. https://doi.org/10.1145/3651890.3672261

Byers, K. (2022, January 19). *An Introduction to Nornir*. https://pynet.twb-tech.com/blog/nornir-an-introduction.html

Chen, J. (2023). Model Algorithm Research based on Python Fast API. *Frontiers in Science and Engineering*, *3*(9), 7–10. https://doi.org/10.54691/fse.v3i9.5591

Chen, S., Ahmmed, S., Lal, K., & Deming, C. (2020). Django Web Development Framework: Powering the Modern Web. *American Journal of Trade and Policy*, *7*(3), 99–106. https://doi.org/10.18034/ajtp.v7i3.675

Choudhury, S. (2024). Zero-Touch Slicing: Revolutionizing 5G Network Management through AI and Automation. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *10*(6), Article 6. https://doi.org/10.32628/CSEIT2410612437

Clemm, A., Ciavaglia, L., Z. Granville, L., & Tantsura, J. (2022). *Intent-Based Networking—Concepts and Definitions* (No. RFC9315; p. RFC9315). RFC Editor. https://doi.org/10.17487/RFC9315

Cobern, W., & Adams, B. (2020). Establishing survey validity: A practical guide. *International Journal of Assessment Tools in Education*, *7*(3), 404–419. https://doi.org/10.21449/ijate.781366

Daraojimba, E. C., Nwasike, C. N., Adegbite, A. O., Ezeigweneme, C. A., & Gidiagba, J. O. (2024). COMPREHENSIVE REVIEW OF AGILE METHODOLOGIES IN PROJECT

MANAGEMENT. *Computer Science & IT Research Journal*, *5*(1), Article 1. https://doi.org/10.51594/csitrj.v5i1.717

Datta, A., Imran, A. T. M. A., & Biswas, C. (2023). Network Automation: Enhancing Operational Efficiency Across the Network Environment. *ICRRD Quality Index Research Journal*, *4*(1). https://doi.org/10.53272/icrrd.v4i1.1

DiCesare, M. (2025, January 16). *The 5 Stages of the Agile Software Development Lifecycle*. https://www.mendix.com/blog/agile-software-development-lifecycle-stages/

Drosos, G.-P., Sotiropoulos, T., Alexopoulos, G., Mitropoulos, D., & Su, Z. (2024). When Your Infrastructure Is a Buggy Program: Understanding Faults in Infrastructure as Code Ecosystems. *Reproduction Package for Article: "When Your Infrastructure Is a Buggy Program: Understanding Faults in Infrastructure as Code Ecosystems*, *8*(OOPSLA2), 359:2490-359:2520. https://doi.org/10.1145/3689799

Dudovskiy, J. (2023). *Positivism—Research Methodology*. https://research-methodology.net/research-philosophy/positivism/

Dugbartey, A. N., & Kehinde, O. (2025). Optimizing project delivery through agile methodologies: Balancing speed, collaboration and stakeholder engagement. *World Journal of Advanced Research and Reviews*, *25*(1), 1237–1257. https://doi.org/10.30574/wjarr.2025.25.1.0193

Edelman, J., Lowe, S. S., & Oswalt, M. (2023). *Network Programmability and Automation*.

Elsevier, S. (2022, November 17). Why is data validation important in research? | Elsevier. *Elsevier Author Services - Articles*. https://scientific-

publishing.webshop.elsevier.com/research-process/why-is-data-validation-important-in-research/

Farayola, O. A., Hassan, A. O., Adaramodu, O. R., Fakeyede, O. G., & Oladeinde, M. (2023). CONFIGURATION MANAGEMENT IN THE MODERN ERA: BEST PRACTICES, INNOVATIONS, AND CHALLENGES. *Computer Science & IT Research Journal*, *4*(2), Article 2. https://doi.org/10.51594/csitrj.v4i2.613

FastAPI. (2024). *Tutorial—User Guide—FastAPI*. https://fastapi.tiangolo.com/tutorial/

Gargano., P. (2020, December 20). *Configuration Management Tools > Network Automation | Cisco Press*. https://www.ciscopress.com/articles/article.asp?p=3100057&seqNum=3

Gondhalekar, B., Manna, H., Kothi, S., & Borsae, D. B. (2024). Network Automation With Multithreading Using GNS3 and Netmiko. *Indian Journal of Computer Science*, *9*(2), 18. https://doi.org/10.17010/ijcs/2024/v9/i2/173860

Gooley, J., Preston, H., & Research, F. (2023). *Scale Your Network Automation*. Forrester. https://www.forrester.com/report/scale-your-network-automation/RES179478

Gottfried, J. (2024). Practices in Data-Quality Evaluation: A Large-Scale Review of Online Survey Studies Published in 2022. *Advances in Methods and Practices in Psychological Science*, *7*(2), 25152459241236414. https://doi.org/10.1177/25152459241236414

Gulati, S., Tyagi, A., & Goel, P. K. (2024). Security Automation and Orchestration in the Cloud: In P. K. Goel, H. M. Pandey, A. Singhal, & S. Agarwal (Eds.), *Advances in Information Security, Privacy, and Ethics* (pp. 24–47). IGI Global. https://doi.org/10.4018/979-8-3693-3249-8.ch002

Huawei, T. (2024). IP Network Digital Map Whitepaper. *Huawei Technologies Co., Ltd.* https://carrier.huawei.com/~/media/cnbgv2/download/products/networks/ip-network-digital-map-white-paper-en-2024.pdf

Imarc. (2024). *Network Automation Market Size, Share and Trends Report 2033*. https://www.imarcgroup.com/network-automation-market

Infonetica. (2024). *5 Essential Ethical Guidelines for Accurate Research in 2024*. https://www.infonetica.net/articles/accurate-ethical-research-guidelines

John, S. (2021). Technology Governance: Minding and Closing the Gaps in Africa. *African Journal of Governance and Development (AJGD), 10*(2), 375–389. https://doi.org/10.36369/2616-9045/2021/v10i2a5

Juniper Networks. (2022). *Junos® OS Chef for Junos OS Getting Started Guide*.

Katariya, H., Gedam, M., Lolage, R., Patil, S., & Shrivastav, U. (2025). *Comparative Analysis of Configuration Management Tools: Chef vs. Ansible, SaltStack, and Puppet*.

Kentik. (2025, May 1). *Network Automation: Enhancing Performance and Reducing Complexity*. Kentik. https://www.kentik.com/kentipedia/network-automation/

Khan, D. A., Hossain, D. M., & Amin, D. S. (2023). *Sampling methods*. https://oercollective.caul.edu.au/customer-insights/chapter/sampling-methods/

Kibuacha, F. (2021, April 6). *How to Determine Sample Size for a Research Study—GeoPoll*. GeoPoll. https://www.geopoll.com/blog/sample-size-research/

Killeen, R. (2025, May 11). Workbook 05 Ansible Configuration for Cisco and Juniper Loops and With Statements. *RichardKilleen*. http://richardkilleen.co.uk/blog/network-

automation/workbook-05-ansible-configuration-for-cisco-and-juniper-loops-and-with-statements/

Klimai, P. (2022). *Day One: Automating Junos® with Salt*. https://www.juniper.net/documentation/en_US/day-one-books/DO_Automating_SALT.pdf

Kumar, M., & Nandal, D. R. (2024a). Python's Role in Accelerating Web Application Development with Django. *International Research Journal on Advanced Engineering and Management (IRJAEM)*, *2*(06), Article 06. https://doi.org/10.47392/IRJAEM.2024.0307

Kumar, M., & Nandal, R. (2024b). Role of Python in Rapid Web Application Development Using Django. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4751833

Landau, P. (2022, March 7). *The 12 Agile Principles: Definitions & How to Use Them*. ProjectManager. https://www.projectmanager.com/blog/agile-principles

Liu, X., Wang, C., Chen, D.-Z., & Huang, M.-H. (2022). Exploring perception of retraction based on mentioned status in post-retraction citations. *Journal of Informetrics*, *16*(3), 101304. https://doi.org/10.1016/j.joi.2022.101304

Mahant, K., & Kalapparambath, J. P. (2024). Exploring the Role of Network Automation in Enterprise Network Scalability and Efficiency. *Educational Administration: Theory and Practice, 30*(11), Article 11. https://doi.org/10.53555/kuey.v30i11.9561

Maptionnaire. (2022, June 17). *12 Best Practices in Survey Design to Try Right Now*. https://www.maptionnaire.com/blog/12-best-practices-in-survey-design

McCombes, S. (2019, September 19). Sampling Methods | Types, Techniques & Examples. *Scribbr*. https://www.scribbr.com/methodology/sampling-methods/

Milne, A. (2022, June 29). An Ultimate Guide on Agile Scrum Methodology? *Net Solutions*. https://www.netsolutions.com/insights/what-is-scrum-development-agile-scrum-methodology/

Mohammad, I. (2024). A Guide for Positivist Research Paradigm: From Philosophy to Methodology. *Idealogy Journal*, *9*(2). https://doi.org/10.24191/idealogy.v9i2.596

Mohammed, A. (2024, December 2). *Industry News 2024 Innovation Is Shaping the Future of Network Engineering*. ISACA. https://www.isaca.org/resources/news-and-trends/industry-news/2024/innovation-is-shaping-the-future-of-network-engineering

Montanari. (2020). *carlmontanari/scrapli—Cisco Code Exchange*. Cisco DevNet Code Exchange. https://developer.cisco.com/codeexchange/github/repo/carlmontanari/scrapli/

Muhammad, T., & Munir, M. (2023). Network Automation. *European Journal of Technology*, *7*(3), 23–42. https://doi.org/10.47672/ejt.1547

Napalm. (2024). *Welcome to NAPALM's documentation! —NAPALM 3 documentation*. https://napalm.readthedocs.io/en/latest/

Network Journey. (2024, July 27). *The Power of Python in Networking: Revolutionizing Network Automation and Scripting – Network Journey*. https://networkjourney.com/the-power-of-python-in-networking-revolutionizing-network-automation-and-scripting/

Njah, Y., Leivadeas, A., Violos, J., & Falkner, M. (2023). Toward Intent-Based Network Automation for Smart Environments: A Healthcare 4.0 Use Case. *IEEE Access*, *11*, 136565–136576. https://doi.org/10.1109/ACCESS.2023.3338189

Ojha, A. K. (2024). Revolutionizing Enterprise Network Management: The Role of Ai-Driven Solutions in Modern Computer Networking. *Journal of Electronics,Computer Networking and Applied Mathematics*, *44*, 1–9. https://doi.org/10.55529/jecnam.44.1.9

Opere, B. (2023, November 13). *(23) The Pivotal Role of Python in Network Automation: A Comprehensive Exploration | LinkedIn*. https://www.linkedin.com/pulse/pivotal-role-python-network-automation-comprehensive-brandon-opere-mldwf/

Osei-Wusu, F., Asiedu, W., Asamoah, K. F., Muntaka, S. A., Yeboah, D., & Sarfo, E. A. (2025). Automating Network Programmability and Backup on Cisco Devices Using Python and Netmiko Library: A Case Study of Komfo Anokye Teaching Hospital LAN. *Journal of Computing Research and Innovation*, *10*(1), Article 1. https://doi.org/10.24191/jcrinn.v10i1.488

Özdoğan, E., Ceran, O., & Üstündağ, M. T. (2023). Systematic Analysis of Infrastructure as Code Technologies. *Gazi University Journal of Science Part A: Engineering and Innovation*, *10*(4), 452–471. https://doi.org/10.54287/gujsa.1373305

Paramiko. (2024). *Welcome to Paramiko! — Paramiko documentation*. https://www.paramiko.org/

Preeti Tupsakhare. (2019). *Python for Automation and Scripting: Streamlining Operations and Increasing Efficiency*. https://doi.org/10.5281/ZENODO.13918609

Puppet. (2024). *What is Infrastructure as Code (IaC)? Best Practices, Tools, Examples & Why Every Organization Should Be Using It | Puppet*. https://www.puppet.com/blog/what-is-infrastructure-as-code

PyNetlabs. (2023, January 19). *What is NETCONF (Network Configuration Protocol)?* https://www.pynetlabs.com/what-is-netconf/

Rahman, A., & Muktadir, Md. G. (2021). SPSS: An Imperative Quantitative Data Analysis Tool for Social Science Research. *International Journal of Research and Innovation in Social Science*, *05*(10), 300–302. https://doi.org/10.47772/IJRISS.2021.51012

Rakissaga, W. A. O., Omar, H. H., & Kouraogo, P. J. (2025). Software Defined Networks: Strengths, Weaknesses, and Resilience to Failures. *Engineering*, *17*(01), 19–29. https://doi.org/10.4236/eng.2025.171002

Ramakrishnan, M. (2022, December 21). What is PyCharm? How is it Useful for Python Development? *Emeritus Online Courses*. https://emeritus.org/blog/coding-what-is-pycharm/

Ramesh, Logeshwaran, J., & Kumar, A. P. (2023). The Smart Network Management Automation Algorithm for Administration of Reliable 5G Communication Networks. *Wireless Communications and Mobile Computing*, *2023*, 1–13. https://doi.org/10.1155/2023/7626803

Rashid, A. (2023, June 28). *Research Philosophy: Positivism, Interpretivism, and Pragmatism*. https://limbd.org/research-philosophy-positivism-interpretivism-and-pragmatism/

Redhat. (2024). *Red Hat | Ansible Automation Platform | Features and Benefits*. https://www.redhat.com/en/technologies/management/ansible/features

Reintech. (2024, March 2). *Chef Compliance: Ensuring and Enforcing Security Standards Across Your Nodes | Reintech media*. https://reintech.io/blog/chef-compliance-security-standards-enforcement

Saeik, F., Avgeris, M., Spatharakis, D., Santi, N., Dechouniotis, D., Violos, J., Leivadeas, A., Athanasopoulos, N., Mitton, N., & Papavassiliou, S. (2021). Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. *Computer Networks*, *195*, 108177. https://doi.org/10.1016/j.comnet.2021.108177

Saivic. (2024). *Why Africa needs automation now—April 2025—SA Instrumentation & Control*. https://www.instrumentation.co.za/24171r

Saleem, S., Asim, M. N., Elst, L. V., & Dengel, A. (2023). FNReq-Net: A hybrid computational framework for functional and non-functional requirements classification. *Journal of King Saud University - Computer and Information Sciences*, *35*(8), 101665. https://doi.org/10.1016/j.jksuci.2023.101665

Salt, P. (2024a). *Salt.modules.netbox*. https://docs.saltproject.io/en/3006/ref/modules/all/salt.modules.netbox.html

Salt, P. (2024b). *Salt.states.firewalld*. https://docs.saltproject.io/en/3006/ref/states/all/salt.states.firewalld.html

Salt, P. (2024c). *Salt.states.netntp*. https://docs.saltproject.io/en/3006/ref/states/all/salt.states.netntp.html

Santyadiputra, G. S., Listartha, I. M. E., & Saskara, G. A. J. (2021). The effectiveness of Automatic Network Administration (ANA) in network automation simulation at Universitas Pendidikan Ganesha. *Journal of Physics: Conference Series*, *1810*(1), 012028. https://doi.org/10.1088/1742-6596/1810/1/012028

Saroja, S., & Haseena, S. (2023). Functional and Non-Functional Requirements in Agile Software Development. In *Agile Software Development* (pp. 71–86). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119896838.ch5

Sarrigiannis, I., Antonopoulos, A., Ramantas, K., Efthymiopoulou, M., Contreras, L. M., & Verikoukis, C. (2022). Cost-Aware Placement and Enhanced Lifecycle Management of Service Function Chains in a Multidomain 5G Architecture. *IEEE Transactions on Network and Service Management*, *19*(4), 5006–5020. https://doi.org/10.1109/TNSM.2022.3187314

Sharma, A. (2025, May 19). Top 10 Pitfalls to Avoid in Network Automation. *Motadata*. https://www.motadata.com/blog/pitfalls-in-network-automation/

Sheth, P., Guo, R., Cheng, L., Liu, H., & Candan, K. S. (2023). Causal Disentanglement for Implicit Recommendations with Network Information. *ACM Transactions on Knowledge Discovery from Data*, *17*(7), 1–18. https://doi.org/10.1145/3582435

Singh, P. (2023). *AI-Driven Personalization in Telecom Customer Support: Enhancing User Experience and Loyalty* (SSRN Scholarly Paper No. 5218986). Social Science Research Network. https://doi.org/10.2139/ssrn.5218986

Stancu, S. N., Shevrikuko, A., & Rueda, D. G. (2019). Evolving CERN's Network Configuration Management System. *EPJ Web of Conferences*, *214*, 08015. https://doi.org/10.1051/epjconf/201921408015

Taherdoost, H. (2021). *Data Collection Methods and Tools for Research; A Step-by-Step Guide to Choose Data Collection Technique for Academic and Business Research Projects*

(SSRN Scholarly Paper No. 4178676). Social Science Research Network. https://papers.ssrn.com/abstract=4178676

Thomas, L. (2023, June 22). Stratified Sampling | Definition, Guide & Examples. *Scribbr*. https://www.scribbr.com/methodology/stratified-sampling/

Thota, P. K. (2025). *Responsible Automation: Ethical Dimensions of Self-Healing Cloud Infrastructure*.

Uchenna Joseph Umoga, Enoch Oluwademilade Sodiya, Ejike David Ugwuanyi, Boma Sonimitiem Jacks, Oluwaseun Augustine Lottu, Obinna Donald Daraojimba, & Alexander Obaigbena. (2024). Exploring the potential of AI-driven optimization in enhancing network performance and efficiency. *Magna Scientia Advanced Research and Reviews*, *10*(1), 368–378. https://doi.org/10.30574/msarr.2024.10.1.0028

Vasudevan, S. (2025, January 24). *How to Leverage Netmiko Python for Network Automation*. https://blog.cloudmylab.com/netmiko-python-for-network-automation

Wang, X., Zhao, M., & He, H. (2018). Reverse Logistic Network Optimization Research for Sharing Bikes. *Procedia Computer Science*, *126*, 1693–1703. https://doi.org/10.1016/j.procs.2018.08.108

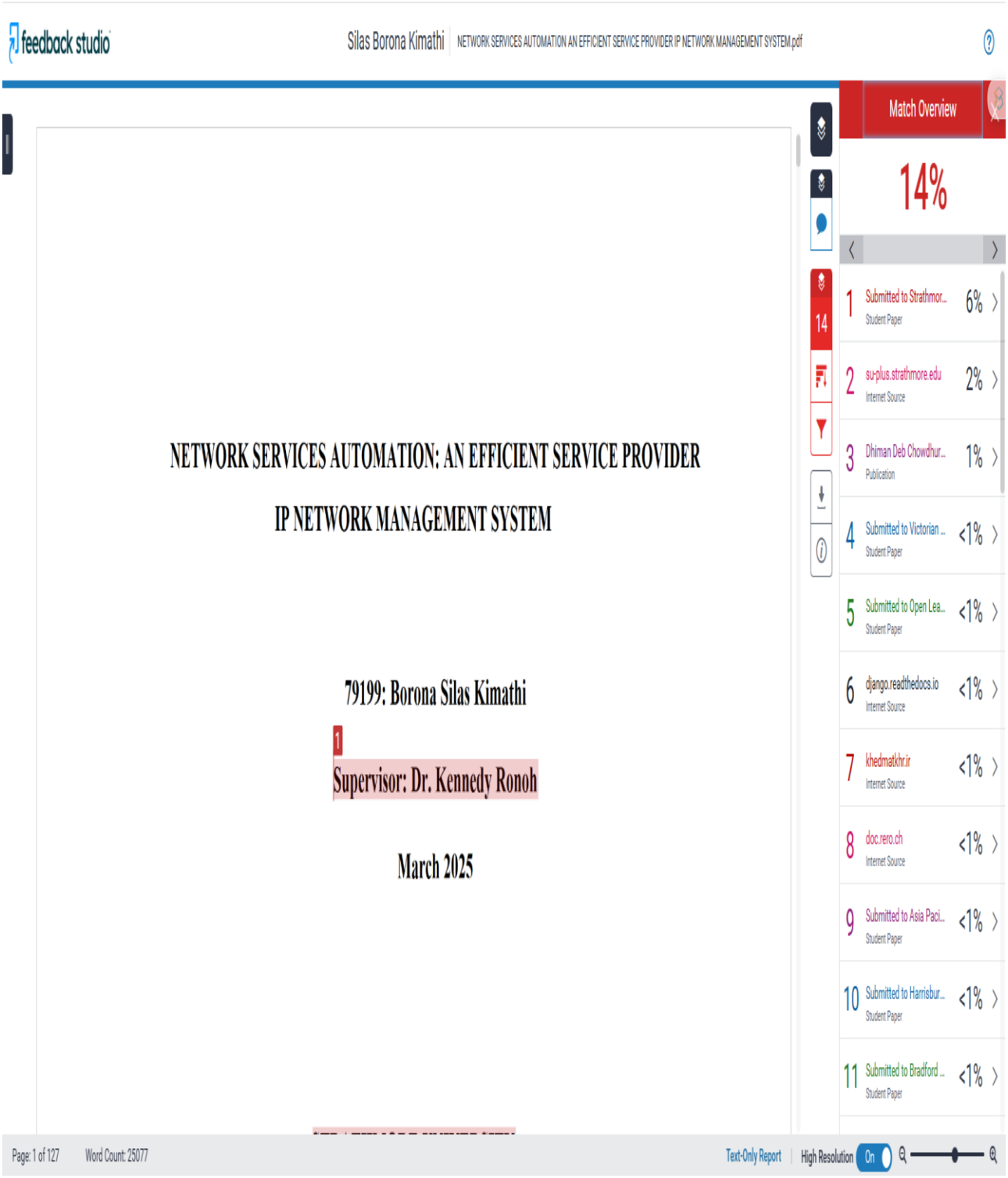
Whizlabs. (2024). *Dharmalingam N, Author at Whizlabs Blog*. https://www.whizlabs.com/blog/author/dharmalingam/

Zhang, H., & Quan, W. (2022). Networking Automation and Intelligence: A New Era of Network Innovation. *Engineering*, *17*, 13–16. https://doi.org/10.1016/j.eng.2021.06.019

Zignuts, T. (2025, March 12). *Django—Build Scalable & Secure Web Applications*. https://www.zignuts.com/blog/django

Appendices

Appendix A: Turnitin Report



Appendix B: Ethical Approval

Strathmore UNIVERSITY

27th November 2024

Mr Borona Silas,
silas.borona@strathmore.edu

Dear Mr Borona,

RE: Network Automation: Efficient Service Provider IP Network Management

This is to inform you that SU-ISERC has reviewed and **approved** your above SU-**masters** proposal. Your application reference number is **SU-ISERC2394/24. The approval period is from 27th November 2024 to 26th November 2025.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

Appendix C: Introduction Letter

Date: ……

Network Engineer's Name: …….

RE: ADMINSTRATION OF QUESTIONNAIRES

I hope this message finds you well. My name is Silas Kimathi Borona. I am a postgraduate student pursuing a Master of Science degree in Telecommunication Innovation at Strathmore University. I am conducting a research study focused on understanding the challenges service providers face in managing current network infrastructures. I would be immensely grateful if you could take a moment to respond to the attached questionnaire, as your insights would provide invaluable contributions to my research.

I want to assure you that all responses will be used solely for academic purposes and treated with the utmost confidentiality. The information gathered will contribute significantly to my research findings and may help inform future strategies for improving network management practices in the industry. Thank you very much for considering this request; your expertise would greatly enhance the quality of my study.

Regards,

Borona Silas Kimathi

Student Number 79199

Strathmore University

School of Computing and Engineering Sciences

Appendix D: Questionnaire

Kindly take some time and answer the following questions.

1. Do you work in the telecom sector? Yes [] No []
2. Sex Male [] Female []
3. Job Position.

   Network Planning Engineer []

   Network Integration Engineer []

   Support Engineer []

   Other []
4. What's your current network management practices?

   Manual []

   Somewhat Manual []

   Automated []

   Somewhat Automated []

   Not Sure []
5. In your own opinion, is the current way of service network management scalable secure?

   Network management is scalable and secure []

   Network management is a bit scalable and secure []

   Network management is fully scalable and secure []

   Network management is not secure and scalable []
6. What specific automation tools API or frameworks have you implemented in your network management processes?

   Redhat Hat Ansible Automation Platform []

   Puppet []

   Chef []

   SaltStack Enterprise []

Napalm []

Netmiko []

Nornir []

SolarWinds Network Automation manager []

Other []

7. What is the pain point you incur while executing you day to day networking role?

   Time-consuming manual configuration []

   Troubleshooting network issues []

   Managing network performance []

   Scaling network infrastructure []

   Maintaining compliance and documentation []

   Coordinating with other IT teams []

   Other []

8. What are some of the network processes you would like to be automated?

   Configuration manager []

   Device inventory and lifecycle management []

   Network Monitoring and Alerts []

   Service Provisioning []

   Compliance and Security Hardening []