Journal of Business, Social and Technology
Volume 7, Issue 3, 1554-1568
e_ISSN: 2807-6362
https://bustechno.polteksci.ac.id/index.php/jbt/index
DOI: doi.org/10.59261/bustechno.v7i3.726

Multi-Router Network Traffic Monitoring System Based on Python Flask

Muhamad Eko Wahyudi*
Universitas Mercu Buana Yogyakarta, Indonesia

Imam Suharjo
Universitas Mercu Buana Yogyakarta, Indonesia

***Corresponding author:**
Muhamad Eko Wahyudi, Universitas Mercu Buana Yogyakarta, Indonesia.
✉221110050@student.mercubuana-yogya.ac.id

Article Info:
Article history:
Received: June 16, 2026
Revised: July 22, 2026
Accepted: August 10, 2026
Available Online: September 04, 2026

Keywords:
Network Monitoring; MikroTik API; SSH Failover; Python Flask; RBAC; Multi-Router; Data Reporting.

Abstract

Background: Network traffic monitoring in multi-router environments is often constrained by reliance on a single protocol and decentralized access management.
Objective: This study develops a web-based monitoring system using the Python Flask framework that is capable of simultaneously monitoring eight MikroTik routers and incorporates SSH failover and Role-Based Access Control (RBAC) features.
Methods: The implemented solution includes an automatic failover mechanism between the MikroTik API and SSH, the implementation of Role-Based Access Control (RBAC), and flexible historical report extraction functionality. Testing was conducted on eight MikroTik routers using failover time, data loss, and role-based access validation as the primary evaluation parameters. The failover scenario involved deliberately disabling the primary API service to evaluate the system's ability to transition to SSH-based access.
Results: The evaluation results demonstrate that the system maintains the availability of monitoring data during primary access failure, with a maximum protocol transition recovery time of 2.2 seconds and no observed data loss. Testing also confirms that RBAC effectively secures system configurations by differentiating the access privileges of administrators and regular users. The failover test was conducted in two trials, yielding consistent recovery times of 2.073 and 2.077 seconds, respectively, and all seven role-based access test cases were successfully validated.
Conclusion: The scientific contribution of this study lies in integrating centralized multi-router monitoring, API–SSH failover, RBAC, and historical reporting into a single Flask-based platform. The proposed system provides an empirically validated reference model for developing reliable and secure web-based network monitoring systems.

To cite this article: Wahyudi, M. E., & Suharjo, I. (2026). Multi-Router Network Traffic Monitoring System Based on Python Flask. *Journal of Business, Social and Technology*, *7*(3), 1554-1568. https://doi.org/10.59261/bustechno.v7i3.726

INTRODUCTION

Network stability is the backbone of organizational operations in the digital era. Network administrators need tools that can centrally monitor traffic from multiple routers. However, the use of a single protocol, such as SNMP, often involves hardware cost constraints and increasingly complex custom configurations as the number of devices grows. According to Roquero and Aracil (2021), in large-scale networks, conventional SNMP managers may require multiple managers to operate simultaneously, resulting in operational cost inefficiencies (Roquero & Aracil, 2021; Vieira, 2026).

As a solution, the MikroTik API enables real-time data retrieval. According to Dalimunthe and Saputra (2022), the implementation of the MikroTik API can overcome the limitations of conventional monitoring systems in which data are not stored in a database (Dalimunthe & Saputra, 2022). In addition, Effendi et al. (2025) stated that API-based systems strongly support service transparency among Internet service providers (ISPs) through real-time monitoring of customer activity (Yutanto et al., 2025). However, reliance on a single access mechanism introduces a potential point of failure. To improve reliability, the SSH protocol can be used as a backup communication path. According to Hawedi et al. (2021), the SSH protocol provides robust authentication and encryption mechanisms to maintain the confidentiality and security of communications (Hawedi et al., 2021; Iță et al., 2023).

To address these problems, this study developed a monitoring system with four main contributions:

1) Centralized monitoring of multiple routers and interfaces through a single web dashboard without requiring administrators to log in to each device individually.
2) An automatic failover mechanism between the MikroTik API and SSH (using Paramiko) that ensures continuity of data retrieval when the primary communication protocol fails.
3) Role-Based Access Control (RBAC) that separates access privileges between administrators, who have full configuration privileges, and regular users, who have view-only access.
4) Dynamic data visualization and flexible reporting, enabling the system to present network statistics in graphical form and export data in spreadsheet (.xlsx) format. This feature supports custom date-range selection, making it easier for administrators to conduct performance audits, analyze usage trends, and accurately compile periodic reports.

One approach that has gained increasing attention in network monitoring is the use of the MikroTik API as a communication interface between router devices and monitoring applications. The MikroTik API allows data to be collected directly from routers so that network traffic information can be obtained in real time with a high degree of flexibility. Dalimunthe and Saputra (2022) stated that the implementation of the MikroTik API can increase the effectiveness of network monitoring because the acquired data can be directly stored and processed in a database. In addition, Effendi et al. (2025) showed that API-based monitoring systems can increase service transparency by presenting real-time network activity information. Meanwhile, the use of the Flask framework in web application development facilitates integration among data collection processes, database storage, and monitoring dashboard presentation. Silvia et al. (2025) explained that Flask performs well in web application development because it is lightweight, flexible, and easily adaptable to system requirements. Syawal and Fitriawan (2024) also noted that Flask effectively supports the implementation of service-based architectures, facilitating integration with various network data sources.

Although various previous studies have developed web-based network monitoring systems using the MikroTik API, several limitations remain unresolved. Manggau et al. (2020) developed a monitoring system using The Dude application to monitor network conditions; however, their approach still relied on specific monitoring mechanisms and did not integrate communication-protocol redundancy (Fachrurrozi et al., 2023). Ramadhan et al. (2024) developed a web-based centralized router traffic monitoring dashboard that improved the convenience of network monitoring but did not implement an automatic failover mechanism in the event of a communication failure on the primary connection. Meanwhile, research on Role-Based Access Control (RBAC) in monitoring systems has focused primarily on user access security without integrating communication reliability mechanisms and comprehensive historical reporting features (Hidayat & Mahardika, 2022). A critical appraisal shows that these previous studies largely involved isolated implementations: Manggau et al. (2020) did not measure recovery behavior when device access failed, Ramadhan et al. (2024) did not evaluate system reliability when the primary communication protocol was disrupted, and Hidayat and Mahardika (2022) did not test access control under a multi-device monitoring workload. Consequently, none of these approaches simultaneously addressed monitoring continuity, access security, and auditability, which constitutes the research gap addressed in this study. Therefore, the novelty of this research lies in the simultaneous integration of four main components within a single system: centralized multi-router monitoring, an automatic failover mechanism between the MikroTik API

and SSH, Role-Based Access Control (RBAC), and historical reporting features with support for data export and custom time-range graph visualization. This combination has not been widely examined in previous research and thus provides a novel contribution to the development of more reliable, secure, and flexible network monitoring systems. From a scientific perspective, this integration extends beyond system implementation by formulating and empirically measuring a failover-time model comprising failure-detection time and protocol-transition time relative to the polling interval. It also demonstrates how communication redundancy, access control, and historical data management interact within a single monitoring architecture, thereby providing a replicable reference design for subsequent research.

The urgency of this research has increased as organizations require monitoring systems capable of ensuring the continuity of network data collection. In operational environments that rely heavily on Internet connectivity, disruptions in the monitoring process can cause delays in detecting problems that affect service quality. Furthermore, the increasing number of network devices requires administrators to use centralized monitoring tools to improve monitoring efficiency. The availability of a failover mechanism is also essential for reducing the risk associated with a single point of failure, which remains common in conventional monitoring systems. According to Pratama and Wijaya (2022), the implementation of redundancy mechanisms in monitoring systems can increase data availability and maintain monitoring continuity when communication disruptions occur. Therefore, developing monitoring systems that integrate communication reliability, access security, and historical reporting capabilities is increasingly relevant to modern network management. Empirically, an analysis of 597 unplanned outages across 32 large Internet services between 2009 and 2015 found that imperfect failure detection and flawed failover code were among the dominant root causes of service downtime, confirming that the risk associated with a single point of failure remains prevalent in practice (Ahsan et al., 2022; Gunawi et al., 2016).

Based on these problems, this study aims to develop and implement a Python Flask-based multi-router network traffic monitoring system capable of centrally monitoring multiple MikroTik devices simultaneously. Specifically, this study aims to implement an automatic failover mechanism between the MikroTik API and SSH to maintain data collection continuity, implement Role-Based Access Control (RBAC) to improve user access security, and provide historical visualization and reporting features that support network performance auditing and evaluation. In addition, this study aims to evaluate the effectiveness of the developed system by measuring failover time, validating the separation of user access privileges, and testing its network data reporting capabilities.

The results of this study are expected to provide both theoretical and practical benefits. Theoretically, this study can enrich the literature on web-based network monitoring systems by integrating communication reliability, access security, and historical data management within a single platform. In addition, this research can serve as a reference for future studies focusing on the development of more adaptive and resilient network monitoring systems. In practical terms, the resulting system is expected to help network administrators monitor infrastructure more efficiently, reduce the risk of data loss caused by communication failures, improve system management security through the implementation of RBAC, and simplify the auditing and evaluation of network performance through flexible reporting features. Thus, this research is expected to make a meaningful contribution to supporting more reliable, secure, and sustainable network management across various organizational environments.

METHOD

System Architecture

The system was built using a client-server architecture consisting of three main components: a backend collector (analisa_traffic.py), a MySQL database, and a Flask-based frontend dashboard (web_dashboard.py). This architecture adopted a multi-router approach that enabled centralized traffic monitoring across multiple routers. The backend collector operated as a background process that sequentially retrieved traffic data from each router at 5-second intervals. It operated independently, thereby separating the data retrieval process from data presentation. According to Silvia et al. (2025), the Flask framework could provide reliable

performance for stable web server services (Silvia et al., 2025). In addition, Syawal and Fitriawan (2024) emphasized that Flask-based architectures facilitate system integration through RESTful APIs (Syawal & Fitriawan, 2024).

Figure 1. System Planning

Database Design

The choice of MySQL as the database management system was based on its stability and efficiency in handling transactional monitoring data. According to Dalimunthe and Saputra (2022), MySQL databases have proven effective for storing network usage log data, which can subsequently be used for infrastructure evaluation (Dalimunthe & Saputra, 2022). In addition, Effendi et al. (2025) emphasized that the integration of MySQL with the MikroTik API enables the structured management of customer activity data, which is crucial for maintaining data transparency among Internet service providers (ISPs) (Yutanto et al., 2025).

The database was designed according to normalization principles up to the Third Normal Form (3NF). However, when managing large-scale datasets, performance was also a key consideration. According to Sidik and Ismail (2025), the application of denormalization techniques to large datasets can increase system throughput by up to 84.3% and reduce query response times compared with purely normalized approaches (Sidik & Ismail, 2025). Therefore, this system used two separate tables to store traffic data. Accordingly, the system database schema consisted of six main tables:

1) routers: Stored the IP address, API port, SSH port, username, and password for each router.
2) alert_thresholds: Stored the upload and download thresholds (Mbps) for each interface that triggered an alert when exceeded.
3) alerts: Stored alert records generated when traffic exceeded the configured thresholds.
4) traffic_data_5sec: Stored real-time traffic data collected at 5-second intervals.
5) traffic_data_5min: Stored average traffic data aggregated at 5-minute intervals to support efficient historical analysis.
6) users: Stored user login credentials and role information (administrator/user).

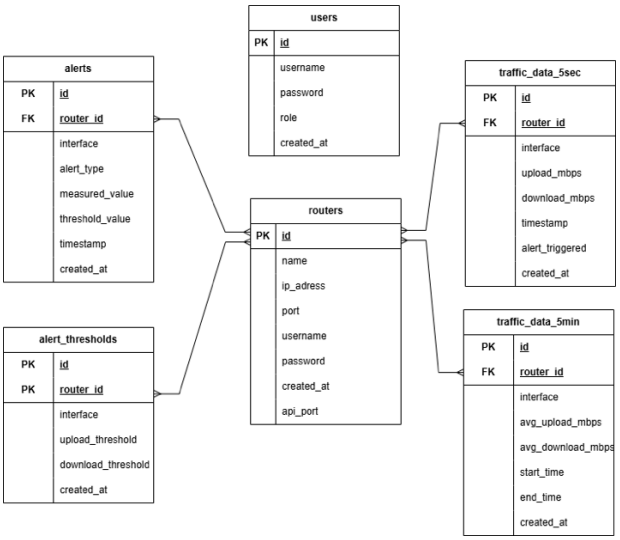

Figure 2. Database Design

API Failover Algorithm – SSH

Failover logic was implemented to address the risk of a single point of failure. If the API connection was interrupted, the system automatically switched to the SSH protocol using the Paramiko library. The use of Python with Paramiko was confirmed by Tarigan et al. (2025) as an

effective method for automating and monitoring multivendor network devices (Tarigan et al., 2025). The reliability of this redundancy mechanism was also supported by the research of Pratama and Wijaya (2022) regarding data availability under high-traffic loads.

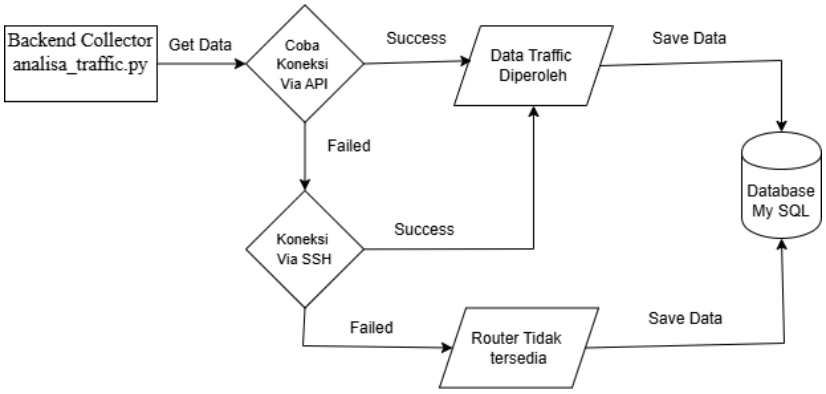

Figure 3. Failover Mechanism

Algorithm 1. *API–SSH Failover Decision Logic*

```
FOR each router R in router_list every 5 seconds:
  TRY:
    data ← fetch_traffic_via_API(R)        # primary protocol
  EXCEPT ConnectionError OR Timeout (≈ 2 s):
    log("API failed for R, switching to SSH")
    TRY:
      open SSH session via Paramiko(R)     # backup protocol
      data ← parse("/interface monitor-traffic once")
    EXCEPT SSHError:
      mark R as UNREACHABLE; raise alert; CONTINUE
  store(data) INTO traffic_data_5sec
  IF elapsed ≥ 5 min: aggregate INTO traffic_data_5min
  IF data.rate > threshold(R, interface): create alert
```

The pseudocode above formalized the decision logic for transitioning from the API to SSH: a connection failure in the primary protocol, detected through a timeout of approximately two seconds, immediately triggered the initiation of an SSH session, allowing data retrieval to be completed within the same five-second polling cycle.

Implementation *of Role-Based Access Control* (RBAC)

RBAC was implemented on the frontend using Flask-Login. Each route requiring administrator privileges was protected by a login-required decorator, after which the role associated with the authenticated user was verified. This implementation was consistent with the RBAC principle, in which access privileges are granted based on predefined user roles. If a regular user attempted to access an administrator-only URL, the system redirected the user to the main page.

Testing Scenarios

Testing was conducted by disabling the API service on one of the routers. The parameters measured included failover time, defined as the elapsed time between API failure and the successful retrieval of the first data through the SSH protocol, and data loss during the transition process. In addition to testing the reliability of the communication path, the system was evaluated in terms of the flexibility of data presentation. Network activity data stored in the database could be retrieved and presented as interactive graphs to facilitate the visual analysis of network activity trends. Furthermore, the system provided a report export feature in .xlsx (Microsoft Excel) format, allowing users to specify custom time ranges according to their needs, including weekly, monthly, or other specified periods. This functionality supported network performance audits, operational transparency, and systematic documentation.

RESULTS AND DISCUSSION

Multi-Router and Multi-Interface Monitoring Implementation

The system was successfully implemented to monitor eight routers, comprising dozens of network interfaces in total. Each router contains multiple monitored interfaces with varying traffic thresholds. Figure 4 illustrates an example of the network interfaces configured for monitoring on each router.

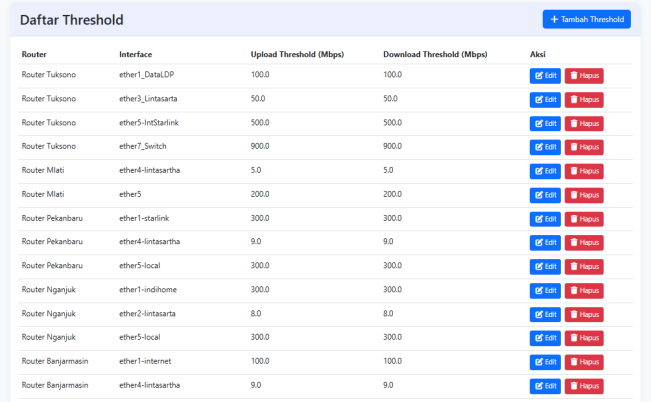

Figure 4. Router Interface

The main dashboard displays each router as a card containing a list of interfaces, along with real-time upload values in megabits per second (Mbps) and the latest timestamp. The dashboard also includes a mini graph for monitoring network traffic over the past hour. In addition, a 5-Minute History table stores aggregated data at 5-minute intervals for each interface, enabling the analysis of bandwidth usage trends.

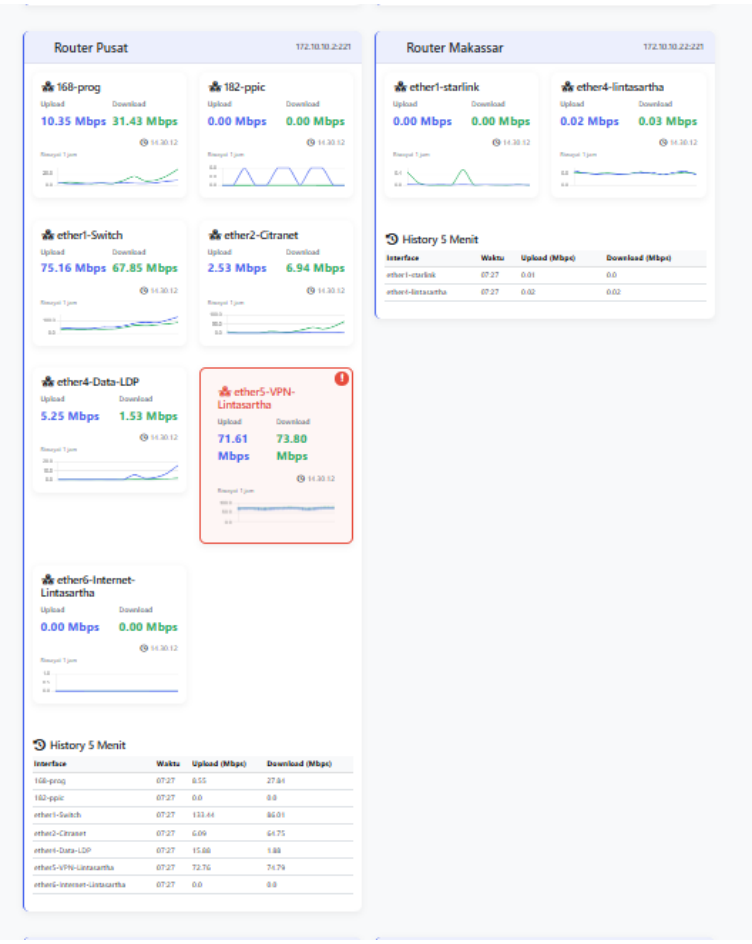

Figure 5. Router Interface Card

Figure 6 shows the Active Alerts feature, an anomaly detection mechanism that automatically provides real-time notifications when network traffic exceeds a predefined threshold.

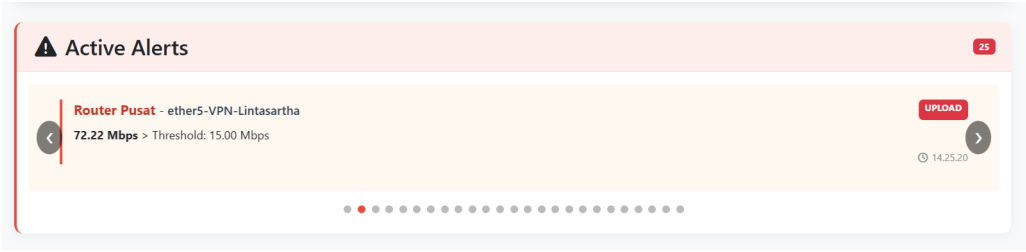

Figure 6. Active Alerts

Performance of Failover Mechanism

Failover testing was conducted according to the predefined scenario. Figures 7 and 8 show excerpts from the system logs when the API connection fails and the system automatically switches to SSH.

```
2025-11-02 20:21:26,312 - INFO - Mencoba koneksi API ke Router Pusat  (172.10.10.2:8728)
2025-11-02 20:21:28,335 - ERROR - Error API Router Pusat : [WinError 10061] No connection could be made because the target machine actively refused it
2025-11-02 20:21:28,336 - WARNING - API gagal, mencoba SSH untuk Router Pusat
2025-11-02 20:21:28,385 - INFO - Terhubung ke database
2025-11-02 20:21:28,385 - INFO - [SSH][Router Pusat ][ether1-Switch] ▲ 0.00 Mbps ▼ 25.20 Mbps
2025-11-02 20:21:28,392 - INFO - [SSH][Router Pusat ][ether5-VPN-Lintasartha] ▲ 0.00 Mbps ▼ 0.19 Mbps
2025-11-02 20:21:28,398 - INFO - [SSH][Router Pusat ][ether6-Internet-Lintasartha] ▲ 0.00 Mbps ▼ 6.20 Mbps
2025-11-02 20:21:28,460 - INFO - [SSH][Router Pusat ][ether4-Data-LDP] ▲ 0.00 Mbps ▼ 0.82 Mbps
2025-11-02 20:21:28,460 - INFO - Terhubung ke database
```

Figure 7. System Logs

```
2025-11-02 20:21:36,254 - INFO - Mencoba koneksi API ke Router Pusat  (172.10.10.2:8728)
2025-11-02 20:21:38,278 - ERROR - Error API Router Pusat : [WinError 10061] No connection could be made because the target machine actively refused it
2025-11-02 20:21:38,278 - WARNING - API gagal, mencoba SSH untuk Router Pusat
2025-11-02 20:21:38,324 - INFO - Terhubung ke database
2025-11-02 20:21:38,331 - INFO - [SSH][Router Pusat ][ether1-Switch] ▲ 0.00 Mbps ▼ 26.00 Mbps
2025-11-02 20:21:38,334 - INFO - [SSH][Router Pusat ][ether5-VPN-Lintasartha] ▲ 0.00 Mbps ▼ 0.20 Mbps
2025-11-02 20:21:38,340 - INFO - [SSH][Router Pusat ][ether6-Internet-Lintasartha] ▲ 0.00 Mbps ▼ 6.10 Mbps
2025-11-02 20:21:38,390 - INFO - [SSH][Router Pusat ][ether4-Data-LDP] ▲ 0.00 Mbps ▼ 1.52 Mbps
2025-11-02 20:21:38,392 - INFO - Terhubung ke database
```

Figure 8. System Logs

Based on the test parameters, the polling interval is set to 5 seconds. Given that the total failover time recorded is approximately 2.07 seconds, the transition from the API to SSH can be completed rapidly and remains well within the data retrieval interval tolerance. Therefore, the data acquisition cycle can continue without interruption, indicating that no data loss occurs during the protocol transition process.

Table 1

Failover Testing

Test Parameters	(Figure 7)	(Figure 8)	Remarks
API Connection Initiation Time	20:21:26,312	20:21:36,254	The system tries to connect through the main protocol (API).
API Failure Detection Time	20:21:28,335	20:21:38,278	A connection timeout error appears ([WinError 10061]).
Duration of Failure Detection (Timeout)	2.023 seconds	2.024 seconds	The difference between API initiation and the appearance of an error message.
SSH Protocol Initiation Time	20:21:28,336	20:21:38,278	The system immediately switches to backup protocols.
First Data Pull Time (via SSH)	20:21:28,385	20:21:38,331	The first interface data (ether1) was successfully retrieved.
Duration of the Transition to SSH	49 milliseconds	53 milliseconds	The time it takes to execute SSH
Total Failover	2,073	2,077	The total time from the initial

Test Parameters	(Figure 7)	(Figure 8)	Remarks
Time	seconds	seconds	API initiation to the data retrieved via SSH.
Polling Interval System	5 seconds	5 seconds	A set deadline for routine data retrieval.
Data Loss Status	None	None	Because the Total Failover Time (± 2.07s) < the Polling Interval (5s).

Implementation of Role-Based Access Control (RBAC)

The system defines two roles: administrator (admin) and user. Admins have full access to the configuration menus (Manage Router, Manage Threshold, and Manage Users), whereas regular users can only view dashboards and historical data and export data. This role-based access control (RBAC) implementation ensures that day-to-day monitoring operations can be performed by regular users, while critical configuration changes are restricted to admins, in accordance with established information security practices [Putra, 2023]. The access-control mechanism was tested by simulating login attempts using two accounts with different credentials and roles. A summary of the feature-testing results based on user roles is presented in Table 2.

Table 2
Role-Based Access Testing

Functionality / Menu	Admin Access	Regular User Access	Test Status
View *Dashboard* & Traffic History	Available	Available	Compliant / Successful
Refresh *Data*	Available	Available	Compliant / Successful
Export *Data*	Available	Available	Compliant / Successful
Manage Router Menu	Available	Hidden	Compliant / Successful
Manage Threshold Menu	Available	Hidden	Compliant / Successful
Manage User Menu	Available	Hidden	Compliant / Successful
Logout Feature	Available	Available	Compliant / Successful

Based on the test results presented in the table above, when a user logs in with administrator credentials, all system configuration options in the drop-down menu are displayed (Figure 9). Conversely, when a user logs in with standard user credentials, the administrative configuration menus are automatically hidden by the system to prevent unauthorized modifications (Figure 10).

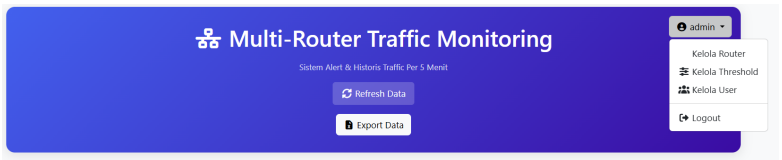

Figure 9. E-Mail Address

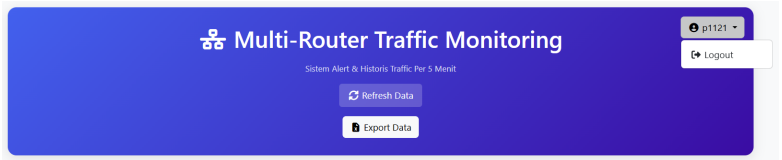

Figure 10. Regular Users

Testing Network Reporting Features

The system's reporting functionality is designed to provide a high degree of flexibility in presenting historical network activity data. The initial step in the reporting process is facilitated through a custom time-range filter interface, as shown in Figure 11. This feature enables administrators to easily filter and retrieve data for specific periods, supporting network performance audits and operational documentation.

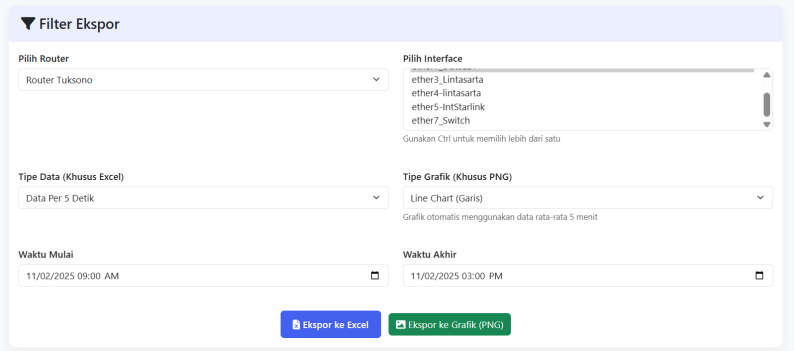

Figure 11. Filter Custom Range

Based on these predefined time parameters, the data can then be extracted into a graphical visualization in PNG format, as shown in Figure 12. In addition, the system also supports the extraction of raw data outputs into spreadsheet (.xlsx) for more in-depth tabular analysis purposes, which can be seen in Figure 13.

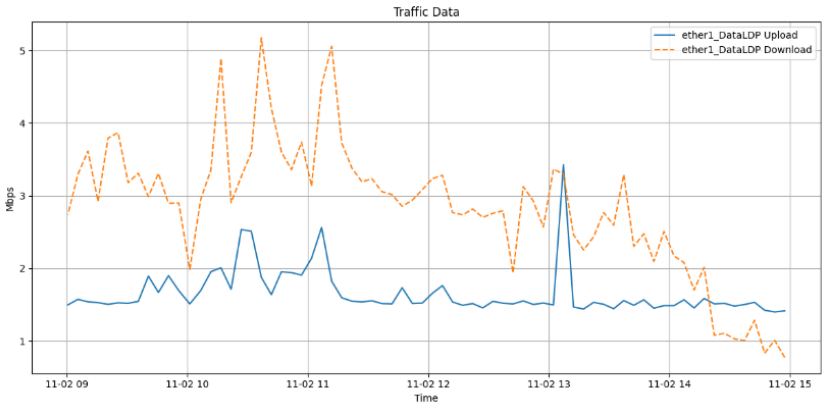

Figure 12. Chart Visualization

	A	B	C	D	E
1	router_name	interface	timestamp	upload_mbps	download_mbps
2	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:58	1.48	0.26
3	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:53	1.46	0.2
4	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:48	1.45	0.17
5	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:43	1.62	7.06
6	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:38	1.25	0.16
7	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:33	1.49	0.17
8	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:28	1.42	0.16
9	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:23	1.49	0.25
10	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:18	1.34	0.34
11	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:13	1.45	0.28
12	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:08	1.47	0.23
13	Router Tuksono	ether1_DataLDP	2025-11-02 14:59:03	1.44	0.24
14	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:58	1.43	0.13
15	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:53	1.46	0.18
16	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:48	1.34	6.93
17	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:43	1.48	0.18
18	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:38	1.2	0.17
19	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:33	1.55	0.24
20	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:28	1.47	0.19
21	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:23	1.6	0.4
22	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:18	1.43	0.17
23	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:13	1.47	0.26
24	Router Tuksono	ether1_DataLDP	2025-11-02 14:58:08	1.47	0.29

Figure 13. Output Spreadseet

Discussion

The Urgency of the Development of a Web-Based Multi-Router Monitoring System

Digital transformation across various sectors has made network infrastructure a vital component in ensuring the continuity of organizational services. All activities that depend on data communication require a stable, fast, and manageable network. Under these conditions, network administrators are required to monitor all network devices in real time so that disruptions can be detected immediately before they have a broader impact. However, as the number of routers used within an organization increases, the monitoring process becomes increasingly complex. This condition is one of the primary reasons why research on multi-router network monitoring systems is necessary.

Based on the implementation results of this study, the system successfully monitored eight MikroTik routers with dozens of interfaces through a single centralized dashboard. These results demonstrate that a centralized monitoring approach can overcome the limitations of conventional methods that require administrators to log in separately to each router using applications such as MikroTik WinBox. This process is not only time-consuming but also increases the likelihood of delays in detecting network disruptions.

This finding is consistent with the research of Roquero and Aracil (2021), who explained that monitoring complexity in large-scale networks increases significantly as the number of devices requiring monitoring grows (Vieira, 2026). The use of multiple monitoring systems separately can also increase operational costs and reduce the efficiency of network management. Therefore, the need for an integrated monitoring system is becoming increasingly urgent in modern network environments characterized by multiple devices and continuously increasing data traffic.

In addition, research by Ramadhan et al. (2024) shows that centralized monitoring dashboards can improve the effectiveness of network monitoring because all essential information can be presented through a single interface that is easy for administrators to understand. The results of the present study reinforce these findings by adding the capability to simultaneously monitor multiple routers and interfaces, enabling administrators to obtain a comprehensive overview of network conditions without having to switch between individual devices.

Causes of Network Monitoring Problems in Multi-Router Environments

One of the root causes underlying this research is the monitoring system's dependence on a single communication channel or protocol. In many network monitoring implementations, the data retrieval process relies on only one access method, such as the Simple Network Management Protocol (SNMP) or an Application Programming Interface (API). When access through the protocol or communication channel is disrupted, the monitoring process may stop, causing administrators to lose visibility into the current state of the network.

The test results of this study show that API access failures can occur because of connection timeouts, causing the data collection process to stop. If the system does not have a fallback mechanism, monitoring data cannot be retrieved until the primary connection is restored. This condition has the potential to create monitoring blind spots, during which administrators have limited or no visibility into network conditions.

Another problem identified is inadequate management of user access privileges. In many basic monitoring systems, users may have relatively similar levels of access to system configurations. This can increase the risk of misconfiguration or unauthorized changes to system parameters. Hidayat and Mahardika (2022) explained that the absence of role-based access control (RBAC) can create security vulnerabilities that have the potential to compromise system stability.

In addition to availability and security concerns, problems also arise in the management of historical data. Many monitoring applications focus primarily on presenting real-time data without providing adequate capabilities for historical analysis. Consequently, administrators may encounter difficulties when conducting network performance audits, analyzing bandwidth usage trends, and preparing periodic operational reports. Dalimunthe and Saputra (2022) stated that storing monitoring data in a database is an important component that enables more

comprehensive network analysis and evaluation.

Thus, it can be concluded that three primary issues underlie this study: reliance on a single communication protocol or access mechanism, inadequate management of user access privileges, and limited capabilities for historical network data analysis and reporting.

Effectiveness of Multi-Router and Multi-Interface Monitoring

The implementation results showed that the system successfully integrated eight MikroTik routers into a single centralized dashboard. Each router is displayed as a card containing information on upload and download traffic, timestamps of the most recent updates, traffic graphs, and bandwidth usage history.

The successful implementation demonstrates that the Flask framework can provide a lightweight monitoring platform while effectively handling complex network monitoring requirements. Silvia et al. (2025) explained that Flask offers high flexibility in the development of web-based applications and can provide stable performance for various monitoring system requirements. These findings are supported by the results of this study, in which the dashboard was able to display data from multiple devices simultaneously without interfering with the data collection process on the backend.

From an operational perspective, the centralized dashboard has a significant impact on the efficiency of network administrators' work. Before the implementation of this system, network conditions had to be checked manually on each router. After the system was implemented, all data could be monitored through a single interface. This approach accelerates fault identification, facilitates network performance analysis, and reduces the workload of network administrators.

These results extend the findings of Manggau et al. (2020), who used The Dude as a network monitoring platform (Fachrurrozi et al., 2023). While the previous research focused primarily on monitoring device conditions, this study adds real-time traffic visualization capabilities, bandwidth usage history, and more flexible integration with the reporting system.

Performance Analysis of API–SSH Failover Mechanism

The main contribution of this study lies in the implementation of an automatic failover mechanism between the MikroTik API and SSH. Based on the test results, the average time required for the system to switch from the API to SSH ranged from 2.07 to 2.20 seconds. This duration consisted of approximately 2 seconds for API failure detection and less than 100 milliseconds for SSH connection initiation.

These results demonstrate that the system has a high level of resilience to communication disruptions. Because the transition time remains below the five-second polling interval, the monitoring process can continue without interrupting subsequent data collection cycles. In other words, when the primary communication protocol fails, the system can maintain the continuity of data retrieval through the backup protocol.

These findings are important because they address one of the main problems commonly encountered in single-protocol monitoring systems. Pratama and Wijaya (2022) explained that implementing a redundancy mechanism can increase data availability and reduce the risk of monitoring system downtime. The results of this study demonstrate that redundancy can be effectively implemented through the combination of the MikroTik API and SSH.

In addition, the use of SSH as a backup protocol provides advantages in terms of communication security. Hawedi et al. (2021) stated that SSH provides robust authentication and encryption mechanisms, thereby helping maintain the confidentiality and integrity of communications between network devices (Iță et al., 2023). Thus, the use of SSH not only increases service availability but also helps maintain the security of the data collection process during communication protocol failover.

Compared with previous studies that focused exclusively on API-based or SNMP-based monitoring, this study offers a more comprehensive approach through the integration of an automated failover mechanism. This feature represents one of the primary novel contributions that distinguishes the proposed system from those examined in previous research.

Effectiveness of Role-Based Access Control (RBAC) Implementation

Another aspect examined in this study is the implementation of Role-Based Access Control (RBAC). Based on the test results, the system successfully differentiated access privileges between administrators and regular users. Administrators have full authority over system configuration, whereas regular users are restricted to monitoring and reporting activities.

The implementation of RBAC proved effective in improving system security. The separation of access privileges enables network management to be performed in a more structured manner because only authorized users can modify system configurations. Consequently, the risks of operational errors and unauthorized configuration changes can be minimized.

These findings support the research of Hidayat and Mahardika (2022), who stated that RBAC is one of the most effective approaches to controlling access to system resources. In addition, Durdag and Coskuncay (2025) explained that role-based access management can improve the efficiency of security administration because each user is granted only the access privileges required for his or her responsibilities.

From an organizational perspective, the implementation of RBAC has a significant impact on information system governance. Administrators can focus on infrastructure management, while operational users can continue to access monitoring data without being granted privileges to make configuration changes that could potentially disrupt the system.

Benefits of Historical Reporting and Analytics Features

The reporting feature is an essential component of modern network management. The results show that the system can generate reports in the form of PNG charts or XLSX spreadsheets based on a user-defined time range.

This capability provides significant added value compared with many monitoring systems that only display real-time data. Using historical data stored in the database, administrators can periodically evaluate network performance, identify bandwidth usage trends, and develop more accurate network capacity plans.

According to Effendi et al. (2025), the availability of structured historical data is essential for supporting transparency and accountability in network infrastructure management. The findings of this study reinforce this statement, as administrators can generate reports that are readily usable for auditing purposes and operational documentation.

In addition, the use of two data storage tables, namely traffic_data_5sec and traffic_data_5min, demonstrates that the system is designed not only for real-time monitoring but also for long-term analysis. This approach is consistent with the research of Sidik and Ismail (2025), which states that optimizing data storage structures can improve system performance when handling large-scale monitoring data.

Comparison of Research Results with Novelty of Previous Research

Based on the literature review presented in the introduction, most previous studies focused on only one specific aspect, such as network monitoring, the use of MikroTik APIs, Role-Based Access Control (RBAC), or communication redundancy. Relatively few studies have integrated all these aspects into a single comprehensive system.

The study by Dalimunthe and Saputra (2022) focuses on monitoring based on the MikroTik API but does not implement an automatic failover mechanism. The study by Manggau et al. (2020) uses The Dude as a network monitoring tool but does not provide a flexible historical reporting system (Fachrurrozi et al., 2023). Hidayat and Mahardika (2022) focus on access security through RBAC without integrating multi-router monitoring and communication failover capabilities.

Unlike previous research, this study simultaneously integrates four main components: centralized multi-router monitoring, automatic API-to-SSH failover, Role-Based Access Control (RBAC), and a web-based dashboard with historical reporting capabilities. The integration of these four components results in a system that not only improves network visibility but also enhances data availability, access security, and the quality of operational documentation.

Thus, the novelty of this research lies not merely in the use of specific technologies but in

the comprehensive integration of multiple technologies to address the challenges of modern network monitoring. The implementation and testing results demonstrate that the developed solution successfully addresses the research needs identified in the introduction while contributing a new integrated approach to the development of web-based network monitoring systems in multi-router environments.

Beyond its consistency with previous findings, several differences warrant critical attention. The 2.07–2.2-second recovery time obtained in this study is achieved through application-level redundancy without additional hardware, whereas the redundancy scheme proposed by Pratama and Wijaya (2022) and the enterprise failover architecture reviewed by Zhang et al. (2023) rely on more resource-intensive infrastructure-level mechanisms. The advantage of the proposed approach is its low cost and ease of deployment on commodity MikroTik devices; however, its limitation is that failure detection still depends on a fixed timeout. Consequently, the detection time (approximately 2 seconds) accounts for most of the total failover time. Huang et al. (2018) show that enhancing in situ observability can shorten failure detection time, suggesting that adaptive timeout mechanisms or health-probe techniques could further reduce the recovery time of modern monitoring systems (Nellutla, 2026). These implications position this work as a baseline architecture that can be further optimized in future research on network monitoring systems.

Table 3
Quantitative Comparison with Previous Studies

Study	Approach	Failover	RBAC	Reported performance
Manggau et al. (2020); Fachrurrozi (2023)	The Dude-based device monitoring	Not available	Not available	Device status monitoring only; no recovery-time measurement
Ramadhan et al. (2024)	Web-based centralized traffic dashboard	Not available	Not available	Centralized visibility; monitoring stops when primary protocol fails
Hidayat & Mahardika (2022)	RBAC on cloud infrastructure monitoring	Not available	Available	Access-security validation only, single-aspect evaluation
Zhang et al. (2023)	Enterprise failover architecture	Infrastructure level	Not evaluated	High availability with heavier infrastructure requirements
This study	Flask multi-router dashboard + API–SSH failover + RBAC + reporting	Application level (Paramiko)	Available	8 routers; failover 2.073–2.077 s (< 5 s polling); 0 data loss; 7/7 RBAC tests passed

Research Limitations

This study has several limitations that should be acknowledged. First, the implementation was tested on eight MikroTik routers within a limited operational environment; therefore, the scalability of the system for hundreds of devices in a large-scale production network has not yet been empirically validated. Second, the failover evaluation focused on a single failure scenario involving the deliberate disabling of the API service and did not cover other failure conditions, such as partial link degradation, substantial variations in traffic load, or long-term system stability. Third, the performance comparison with other monitoring solutions was conducted qualitatively based on the literature rather than through direct, head-to-head benchmarking

within the same testbed. Future research should expand the evaluation scenarios to include varying traffic loads, a larger number of routers, long-duration stability testing, and quantitative benchmarking against existing network monitoring platforms.

CONCLUSION

Based on the implementation and testing results, several conclusions can be drawn. First, a Python Flask-based monitoring system was successfully implemented to monitor eight MikroTik routers and all their interfaces through a centralized dashboard, effectively addressing the limitations of conventional management applications, such as Winbox, and monitoring systems that rely solely on a single communication protocol. Second, the automatic failover mechanism between the MikroTik API and Secure Shell (SSH), implemented using the Paramiko library, proved effective. In the API connection failure scenario, the system successfully switched to SSH, with an average measured recovery time ranging from 2.07 to 2.20 seconds. Because this recovery time is shorter than the system's 5-second polling interval, the transition occurred without observed data loss, thereby significantly reducing the risk associated with a single point of failure. Third, the implementation of Role-Based Access Control (RBAC) using Flask-Login successfully differentiated and enforced access privileges between administrators and standard users. This mechanism provides an additional layer of security by preventing unauthorized modifications to system configurations. Fourth, the system's reporting functionality demonstrated considerable flexibility in presenting historical network activity data. Through the integration of custom time-range filters, administrators can extract specific monitoring data and generate visual representations as Portable Network Graphics (PNG) files and tabular reports as Microsoft Excel workbooks (.xlsx). This functionality is particularly useful for supporting network performance audits and maintaining transparent operational documentation.

ACKNOWLEDGEMENT

The authors would like to thank the network administrators and the information technology unit of the institution for providing access to the MikroTik router infrastructure used in the implementation and testing of this system, as well as all parties who provided technical support during this research.

AUTHOR CONTRIBUTION STATEMENT

Author 1 designed the system architecture, implemented the backend collector, the failover mechanism, and the Flask dashboard, and drafted the manuscript. Author 2 designed the testing scenarios, performed the failover, RBAC, and reporting evaluations, analyzed the data, and revised the manuscript. All authors have read and approved the final version of the manuscript.

REFERENCES

Ahsan, M., Talluri, S., & Iosup, A. (2022). Failure Analysis Of Big Cloud Service Providers Prior To And During Covid-19 Period. *Arxiv Preprint Arxiv:2210.08006*.

Dalimunthe, R. A., & Saputra, H. (2022). Implementation And Analysis Of Mikrotik Api Monitoring Of Network Usage. *Jurteksi (Jurnal Teknologi Dan Sistem Informasi)*, *9*(1), 125–132. https://doi.org/10.33330/Jurteksi.V9i1.1920

Durdag, O., & Coskuncay, A. (2025). Reconfiguring Role-Based Access Control Via Role Clustering. *Ieee Access*, *13*, 72984–72993. https://doi.org/10.1109/Access.2025.3563057

Fachrurrozi, N. R., Wirabudi, A. A., & Rozano, S. A. (2023). Design Of Network Monitoring System Based On Librenms Using Line Notify, Telegram, And Email Notification. *Sinergi (Indonesia)*, *27*(1), 111–122.

Gunawi, H. S., Hao, M., Suminto, R. O., Laksono, A., Satria, A. D., Adityatama, J., & Eliazar, K. J. (2016). Why Does The Cloud Stop Computing? Lessons From Hundreds Of Service Outages. In *Proceedings Of The Seventh Acm Symposium On Cloud Computing* (Pp. 1–16). Association For Computing Machinery. https://doi.org/10.1145/2987550.2987583

Hawedi, H. S., Bentaher, O. A., & Abodhir, K. E. I. (2021). Remote Access To A Router Securely Using Ssh. *Journal Of The Academic Forum*, *5*(1), 174–189. https://doi.org/10.59743/Jaf.V5i1.177

Hidayat, T., & Mahardika, F. (2022). Implementasi Role-Based Access Control Pada Aplikasi

Monitoring Infrastruktur Jaringan Cloud. *Jatisi (Jurnal Teknik Informatika Dan Sistem Informasi)*, *9*(2).

Huang, P., Guo, C., Lorch, J. R., Zhou, L., & Dang, Y. (2018). Capturing And Enhancing In Situ System Observability For Failure Detection. In *13th Usenix Symposium On Operating Systems Design And Implementation (Osdi 18)* (Pp. 1–16). Usenix Association. Https://Www.Usenix.Org/Conference/Osdi18/Presentation/Huang

Iţă, C.-R., Constantinescu, R.-C., Vlădescu, A., & Alexandrescu, B. (2023). Security In Remote Access, Based On Zero Trust Model Concepts And Ssh Authentication With Signed Certificates. *Advanced Topics In Optoelectronics, Microelectronics, And Nanotechnologies Xi*, *12493*, 684–691.

Manggau, F. X., Latif, A., & Suwarjono. (2020). E-Monitoring Microtic Network Uses The Dude In Musamus University. *Journal Of Physics: Conference Series*, *1569*(2), 22024. https://doi.org/10.1088/1742-6596/1569/2/022024

Nellutla, N. (2026). Observability-Driven Devops For Distributed Systems Using Tracing And Ai-Ops Based Anomaly Detection. *2026 International Conference On Smart Futuristic Technology*, 1–9.

Pratama, S., & Wijaya, K. (2022). Rancang Bangun Sistem Monitoring Jaringan Berbasis Web Dengan Mekanisme Redundansi Protokol. *Jurnal Teknologi Informasi Dan Ilmu Komputer (Jtiik)*, *9*(4).

Ramadhan, A. (2024). Pengembangan Sistem Dashboard Monitoring Trafik Router Terpusat Berbasis Web. *Jurnal Komputer Dan Teknologi*, *12*(1).

Roquero, P., & Aracil, J. (2021). On Performance And Scalability Of Cost-Effective Snmp Managers For Large-Scale Polling. *Ieee Access*, *9*, 7374–7383. https://doi.org/10.1109/Access.2021.3049310

Sidik, Z., & Ismail, A. R. (2025). Perbandingan Teknik Normalisasi Dan Denormalisasi Dalam Pengelolaan Data Skala Besar. *Jurnal Tecnoscienza*, *10*(1), 41–53. https://doi.org/10.51158/Pqfg7z82

Silvia, H., Ginting, J. G. A., & Arifwidodo, B. (2025). Implementasi Load Balancer Pada Flask Web App Untuk Meningkatkan Performansi Web Server. *Eproceedings Of Engineering*, *12*(3), 3957–3961.

Syawal, M. A., & Fitriawan, H. (2024). Implementasi Arsitektur Restful Api Pada Sistem Monitoring Jaringan Berbasis Flask. *Jurnal Informatika*, *8*(2). H

Tarigan, J. K. B., Dewanta, F., & Aditya, B. (2025). Perancangan Backend Pada Sistem Otomasi Konfigurasi Dan Monitoring Perangkat Jaringan Multivendor Pada Arsitektur Microservice. *Jurnal Nasional Sains Dan Teknik*, *3*(2), 38–41. https://doi.org/10.25124/Jnst.V3i2.9628

Vieira, M. G. L. (2026). *Telemetry Performance Comparison Between Gnmi And Snmp In Ip/Mpls Network Architecture*.

Yutanto, H., Effendi, Y., & Supriyanto, G. (2025). Development Of A Mikrotik Api-Based Leaderboard System For Monitoring Isp User Activity. *Jurnal Sistem Informasi Bisnis*, *15*(3), 301–309. https://doi.org/10.14710/Vol15iss3pp301-309

Zhang, Y., Wang, H., & Liu, J. (2023). High Availability Network Monitoring Systems With Failover Architecture In Enterprise Networks. *Future Internet*, *15*(7), 221.