

Implementation of a Telegram Bot for Router Monitoring Using Python and the RouterOS API

Alya Namira*

Computer Science Study Program, Faculty of Mathematics
and Natural Sciences
Universitas Negeri Medan, Medan, Indonesia
alyanamira3010@gmail.com

Adinda Soleha

Computer Science Study Program, Faculty of Mathematics
and Natural Sciences
Universitas Negeri Medan, Medan, Indonesia
adindasoleha64@gmail.com

Dedy Kiswanto

Computer Science Study Program, Faculty of Mathematics
and Natural Sciences
Universitas Negeri Medan, Medan, Indonesia
dedykiswanto@unimed.ac.id

Abstract—Network monitoring plays an important role in maintaining the stability and availability of computer network services. Manual monitoring approaches often cause delays in detecting network faults and responding to disruptions. This study implements a MikroTik router monitoring system that integrates Python, the RouterOS API, a Telegram Bot, and a Streamlit dashboard to support real-time fault detection and administrator notification. The system periodically checks router connectivity status and automatically detects when a router transitions between online and offline states. Upon detecting a status change, the system delivers immediate notifications to administrators through Telegram Bot and records the event in a log file for diagnostic purposes. Router metrics including CPU usage, memory availability, uptime, and interface conditions are displayed through a web-based interactive dashboard, providing administrators with a visual interface for monitoring network health. System testing was conducted by simulating router online and offline conditions. Results show that the system successfully detects router status changes, maintains diagnostic logs, and delivers real-time notifications via Telegram and a web-based monitoring interface. This integration of automated fault detection, mobile notification, and web-based visualization offers an effective approach to improving network monitoring responsiveness.

Keywords—Fault Detection; Network Diagnostics; MikroTik RouterOS; Web-Based Monitoring; Mobile Notification; Real-Time Dashboard



© 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution ShareAlike (CC BY SA) license (<https://creativecommons.org/licenses/by-sa/4.0/>).

I. INTRODUCTION

The rapid development of information and communication technology has increased the demand for computer networks to support data exchange and Internet access. Network administrators are required to ensure that network services

remain stable and available to support various user activities effectively. Therefore, network monitoring has become an essential component in the management of computer network infrastructure [1]–[3].

Network monitoring plays an important role in helping administrators continuously observe the condition of network devices and services. Through monitoring activities, network disruptions can be detected more quickly, thereby reducing the risk of service degradation and improving network reliability [4]–[6].

However, in many implementations, network monitoring is still performed manually, making it difficult for administrators to identify and respond to network problems promptly. Conditions such as downtime, unstable connections, and delayed troubleshooting may reduce service quality and negatively affect user activities [1], [7].

MikroTik RouterOS is one of the most widely used network operating systems for network management because it provides various features, including routing, firewall management, bandwidth control, hotspot services, and network monitoring. In addition, RouterOS supports API integration, allowing monitoring and network automation processes to be implemented more flexibly [8]–[10].

The development of network automation technologies has significantly improved the efficiency of computer network management. By utilizing the Python programming language, device data collection, monitoring processes, and notification delivery can be automated, reducing human errors and improving administrative efficiency [11]–[13].

Furthermore, Python-based frameworks such as Streamlit enable the development of interactive and real-time monitoring dashboards. Visual dashboards help administrators monitor network conditions more effectively through the presentation of information in the form of tables, charts, and status indicators [14], [15].

Several previous studies have utilized Telegram Bot as an automatic notification platform for MikroTik-based network monitoring systems. These studies demonstrated that Telegram

Bot can provide real-time network status information to administrators, enabling faster responses to network disruptions [16].

Nevertheless, most existing studies primarily focus on message-based notification systems and do not provide an integrated visual dashboard for comprehensive network monitoring [20]. Therefore, this study develops a network monitoring system based on MikroTik RouterOS by integrating Python, RouterOS API, Telegram Bot, and a Streamlit dashboard. The proposed system is capable of monitoring router status in real time, delivering automatic notifications through Telegram Bot, and presenting network information through an interactive web-based dashboard.

II. LITERATURE REVIEW

A. Network Monitoring

Network monitoring is the process of observing network devices and services to ensure that all network components operate properly. Monitoring activities are performed to detect network disruptions, evaluate network performance, and assist administrators in troubleshooting network-related issues [1], [16].

B. MikroTik and RouterOS

MikroTik RouterOS is a Linux-based operating system designed for computer network management. RouterOS provides a wide range of features, including routing, firewall management, bandwidth management, hotspot services, user management, and network monitoring, enabling efficient network administration [17].

MikroTik is widely utilized in various network environments due to its capability to manage network traffic, control bandwidth allocation, and enhance network security. Furthermore, RouterOS supports API integration, allowing greater flexibility in network automation and monitoring processes.

Fig. 1 illustrates the implementation of MikroTik RouterOS in a virtualized environment. RouterOS serves as a network operating system that provides routing, firewall management, bandwidth control, and network monitoring functionalities to support effective network administration.

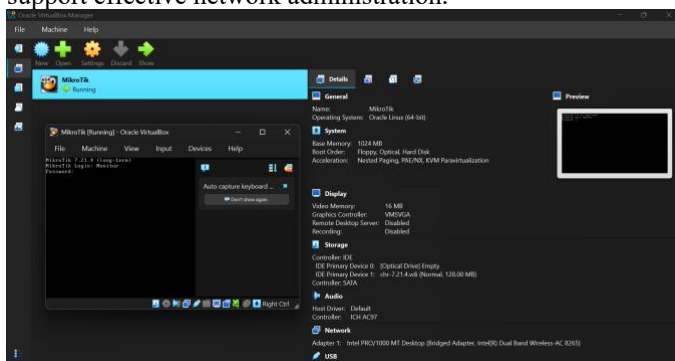


Fig. 1. MikroTik RouterOS Interface in a Virtualized Environment

C. Python and Network Automation

Python is one of the most widely used programming languages in network automation due to its simple syntax and extensive support for networking libraries. Python enables administrators to automate tasks that were previously performed manually, including device data collection, network monitoring, and device configuration [11].

Python-based network automation can improve network management efficiency while reducing the possibility of human error. In addition, the integration of Python with RouterOS API enables automated and real-time network monitoring. Fig. 2 illustrates the application of Python in network automation. Python is utilized for device information retrieval, network monitoring, and RouterOS API integration, enabling network supervision to be performed automatically and in real time.

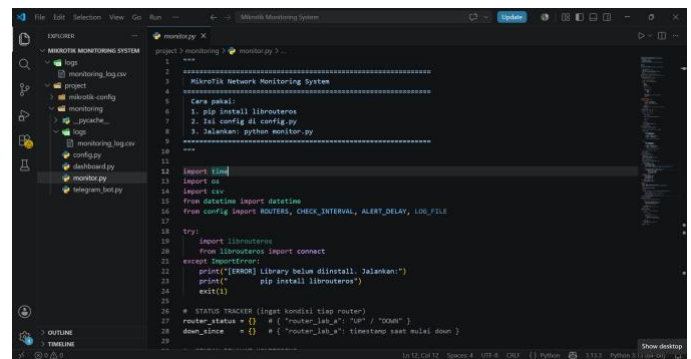


Fig. 2. Network Automation Implementation Using Python

D. Telegram Bot

Telegram Bot is an automated service that can be used to deliver information and notifications to users through the Telegram application. In network monitoring systems, Telegram Bot functions as a communication medium for notifying administrators about device status changes or network disruptions [16].

The use of Telegram Bot allows administrators to receive network status information instantly without manually accessing the monitoring system. Additionally, Telegram Bot can be integrated with various Python-based monitoring applications. Fig. 3 illustrates the process of creating a Telegram Bot using BotFather. The generated bot is utilized to deliver automatic notifications to administrators whenever network disruptions or status changes occur.

E. Streamlit

Streamlit is a Python-based framework that simplifies the development of web applications and interactive dashboards. The framework enables real-time data visualization without requiring complex web development configurations. In network monitoring systems, Streamlit can display device information through tables, charts, and status indicators in an interactive manner. Such visual dashboards help administrators understand network conditions more efficiently and accurately [14],[15]. Fig. 4 shows a monitoring dashboard developed using

the Streamlit framework. The dashboard presents real-time router information, including connection status, CPU usage, memory availability, uptime, and network interface status.

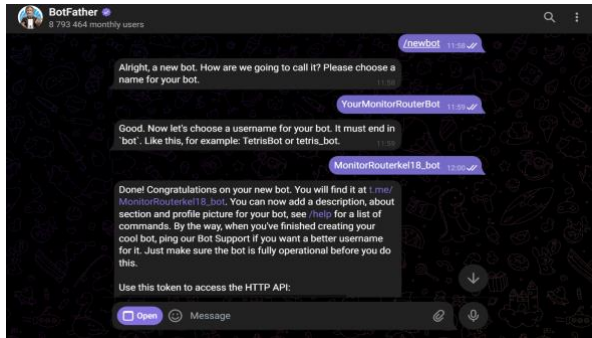


Fig. 3. Telegram Bot Creation Using BotFather

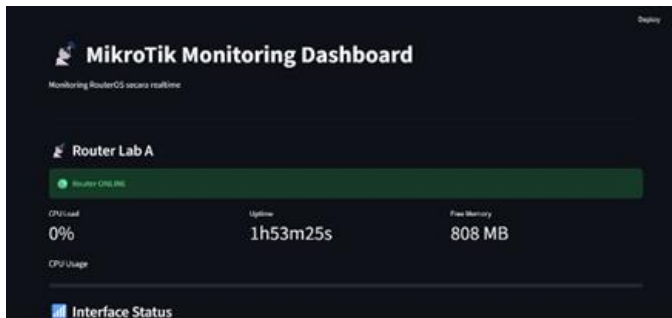


Fig. 4. Streamlit-Based Network Monitoring Dashboard

F. Previous Studies

Numerous studies have investigated MikroTik-based network monitoring systems integrated with Telegram Bot notifications [4]–[6], [16]. These studies indicate that Telegram Bot can effectively provide real-time network status information to administrators. In addition, studies on network automation using Python have demonstrated improvements in network management efficiency and reductions in human error [11]–[13]. Furthermore, Streamlit has been widely utilized for developing interactive dashboards that facilitate information visualization and support real-time data presentation [14], [15].

Pertiwi et al. developed a web-based network logging system for real-time monitoring and analysis by integrating interactive dashboards and Telegram notifications [18]. The study demonstrated that data visualization and real-time monitoring can assist administrators in supervising network activities more effectively.

To better identify the research gap and distinguish the proposed study from previous works, a comparative analysis of related studies is presented in TABLE 1. The comparison highlights the main contributions and limitations of previous studies related to MikroTik monitoring, Telegram-based notifications, and web-based monitoring systems. Through this analysis, the research gap addressed by the proposed system can be more clearly identified.

As presented in TABLE 1, previous studies provide valuable contributions to network management and monitoring, including Telegram-based notifications, remote MikroTik administration,

and web-based monitoring systems. However, these studies generally address specific functionalities separately and do not provide a fully integrated platform that combines real-time monitoring, automated notifications, activity logging, and interactive visualization. In contrast, the proposed system integrates RouterOS API-based monitoring, automated Telegram notifications, activity logging, and Streamlit-based real-time visualization within a single platform. This integration represents the main contribution of the study and provides a more comprehensive solution for network administrators.

Based on the comparative analysis, a research gap exists in the integration of RouterOS API-based monitoring, automatic Telegram notifications, activity logging, and interactive dashboard visualization within a single monitoring platform. Therefore, this study proposes an integrated monitoring system that combines Python, RouterOS API, Telegram Bot, and Streamlit Dashboard. The novelty of the proposed system lies in its ability to provide real-time monitoring, automated fault notifications, monitoring history logging, and web-based visualization through a unified platform.

TABLE 1. Comparison of Related Studies

Study	Main Contribution	Research Gap
Jamil Al Fajar et al. [16]	Telegram Bot for MikroTik monitoring	No dashboard visualization and logging features
Wahyu et al. [5]	Real-time network notifications via Telegram	Focuses only on notification mechanisms
Rahman et al. [6]	Remote MikroTik administration using Telegram	No integrated web-based monitoring dashboard
Pertiwi et al. [18]	Web-based real-time monitoring and logging	Not specifically designed for RouterOS monitoring and Telegram alerts
Proposed Study	Integration of RouterOS API, Telegram Bot, activity logging, and Streamlit dashboard	Provides a unified monitoring platform with real-time visualization and notifications

III. RESEARCH METHOD

This study employs a system implementation approach by integrating the RouterOS API, the Python programming language, a Telegram Bot, and a Streamlit Dashboard to perform real-time monitoring of MikroTik routers. The system is designed to detect changes in router status (UP or DOWN), send automatic notifications via Telegram, and display router condition information through a web-based dashboard.

A. System Design

The proposed monitoring system consists of four main components: a MikroTik router, a Python-based monitoring application, a Telegram Bot, and a Streamlit dashboard. The MikroTik router serves as the source of network information, while the monitoring application communicates with RouterOS through the RouterOS API to collect router data. The collected information is then utilized to generate Telegram notifications and display real-time monitoring results on the dashboard.

Fig. 5 illustrates the workflow of the proposed monitoring system. The process begins with the initialization of system configurations and the list of routers to be monitored. The application then establishes a connection to RouterOS through the API to retrieve information regarding router status, CPU utilization, memory usage, uptime, and interface conditions. The acquired data are analyzed to determine the router status (UP or DOWN). Whenever a status change is detected, the system automatically sends a notification through Telegram and records the event in the monitoring history. Furthermore, router information is displayed in real time on a Streamlit-based dashboard. The monitoring process is executed periodically at predefined intervals, enabling continuous observation of network conditions.

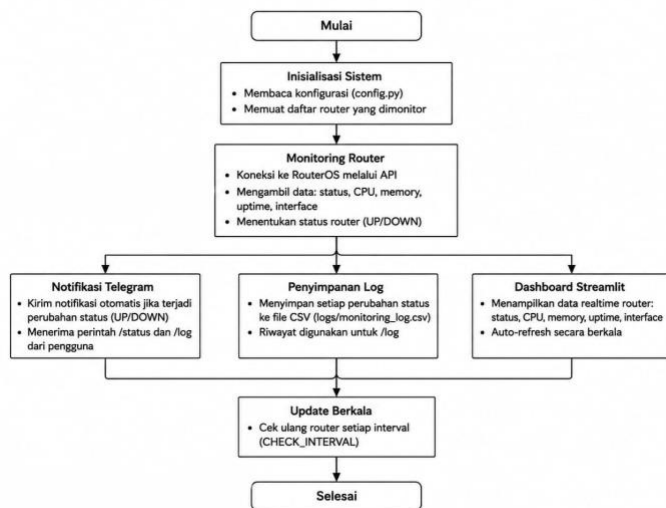


Fig. 5. System Monitoring Workflow

B. RouterOS API Implementation

Communication between the monitoring application and the router is established using the *librouteros* library. The system connects to RouterOS through the API service on port 8728 using a predefined IP address, username, and password. Once the connection is successfully established, the application retrieves information related to router status, CPU utilization, memory capacity, and network interface conditions.

C. Router Monitoring Implementation

The monitoring process is performed periodically using a loop-based mechanism with predefined intervals. The system continuously checks the connectivity status of each registered router. If a router becomes inaccessible for a specified period, its status is changed to DOWN. Conversely, when the connection is successfully re-established, the router status is updated to UP.

In addition to real-time monitoring, the system stores router status changes in a CSV file containing the event timestamp, router name, and router status. This data serve as a monitoring history log and can be retrieved through Telegram commands when required.

D. Telegram Bot Implementation

The Telegram Bot was created using BotFather and integrated into the monitoring application through the Telegram Bot API. The bot is responsible for sending automatic notifications whenever a router status change occurs. Furthermore, users can directly monitor network conditions through the `/status` command to view the current router status and the `/log` command to access monitoring activity records.

E. Router Monitoring Implementation

The monitoring dashboard was developed using the Streamlit framework. The dashboard provides real-time information regarding router conditions, including router connectivity status, CPU utilization, available memory capacity, uptime, and the status of each network interface. Data are updated automatically, allowing users to monitor network performance continuously without manually refreshing the web page.

F. System Testing

System testing was conducted by simulating both operational (UP) and non-operational (DOWN) router conditions. The tests were performed by disabling and re-enabling RouterOS within a virtualization environment. The system performance was evaluated based on its ability to detect router status changes, automatically send Telegram notifications, maintain monitoring history records, and display accurate information on the real-time dashboard.

IV. RESULTS AND DISCUSSION

A. Monitoring System Implementation

The network monitoring system was developed using the Python programming language and integrated with MikroTik RouterOS through the RouterOS API. The system was designed to monitor router conditions in real time and automatically notify administrators whenever network status changes occur.

Fig. 6 is the monitoring system implementation. The implementation consists of three main components: a monitoring module (`monitor.py`), a Telegram Bot module (`telegram_bot.py`), and a Streamlit-based dashboard module (`dashboard.py`). The monitoring module periodically checks router status through the RouterOS API. Monitoring results are then recorded in the `monitoring_log.csv` file as a historical record of system activities.

B. Router Monitoring Implementation

Fig. 7 and Fig. 8 is the monitoring process performed by establishing a connection to the RouterOS API using the *librouteros* library. The system periodically checks router availability to determine whether the router status is UP or DOWN. When the router is successfully accessed, the system identifies the router status as UP. Conversely, if the connection to RouterOS fails, the router is classified as DOWN. To minimize false alarms caused by temporary network

interruptions, the system implements a 10-second alert delay before sending notifications to the administrator. In addition to detecting router status, the system records every status change in the monitoring_log.csv file. The log file contains monitoring timestamps, router names, and router status information.

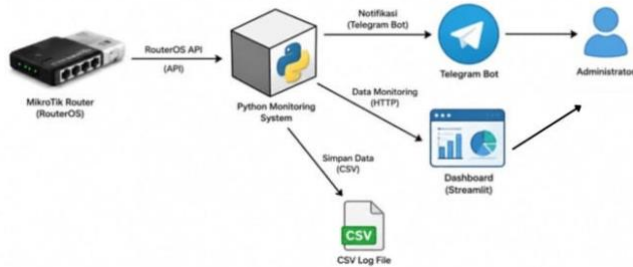


Fig. 6. Python-Based MikroTik Router Monitoring System Architecture

```
=====
MikroTik Monitoring System - AKTIF
Memantau 1 router | interval 10s
=====
[BOT] Polling aktif

[09:30:53] Mengecek semua router...
→ Cek Router Lab A (192.168.100.181)...
[Router Lab A] Terdeteksi DOWN - menunggu 10s sebelum notif...
Selesai. Tunggu 10 detik...

[09:31:06] Mengecek semua router...
→ Cek Router Lab A (192.168.100.181)...
[NOTIF] Router Lab A DOWN
[TELEGRAM] Pesan terkirim
Selesai. Tunggu 10 detik...
```

Fig. 7. Router Monitoring Process Using Python

```
Waktu,Router,Status
2026-05-17 20:54:10,Router Lab A,DOWN
2026-05-17 20:57:34,Router Lab A,DOWN
2026-05-17 21:08:30,Router Lab A,DOWN
2026-06-04 06:43:21,Router Lab A,DOWN
2026-06-04 06:46:48,Router Lab A,UP
```

Fig. 8. Monitoring Log File (CSV)

C. Telegram Bot Implementation

As can be seen in Fig. 9 and Fig. 10, the telegram Bot serves as a notification and communication medium between the monitoring system and network administrators. The system automatically sends notifications whenever the router status changes from ONLINE to OFFLINE or vice versa. In addition to automatic notifications, the Telegram Bot provides several commands that can be used by administrators:

1. /start – Displays information about the monitoring system.
2. /status – Displays the current router status in real time.
3. /log – Displays the five most recent monitoring records.

These features allow administrators to obtain network status information without directly accessing the monitoring system.

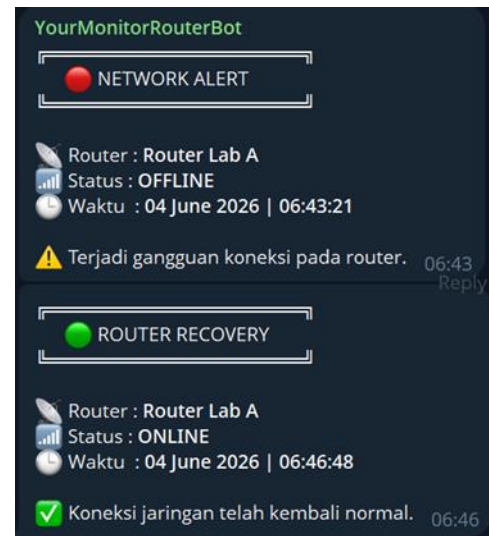


Fig. 9. Telegram Bot Notification



Fig. 10. Telegram Bot /log Command Output

D. Streamlit Dashboard Implementation

The monitoring dashboard which shown in Fig. 11 was developed using the Streamlit framework as the visual interface of the monitoring system. The dashboard displays router information in real time and refreshes automatically every five seconds. When the router is in the ONLINE state, the dashboard presents several important parameters, including:

1. Router status.
2. CPU load
3. Device uptime.

4. Available memory
5. Network interface status.

In addition, the dashboard utilizes visual indicators such as progress bars to represent CPU utilization (Fig. 12). Network interface information is displayed separately, allowing administrators to monitor the condition of each interface individually. When the router becomes inaccessible, the dashboard automatically displays an OFFLINE status to indicate a connectivity issue which can be seen in Fig. 13.

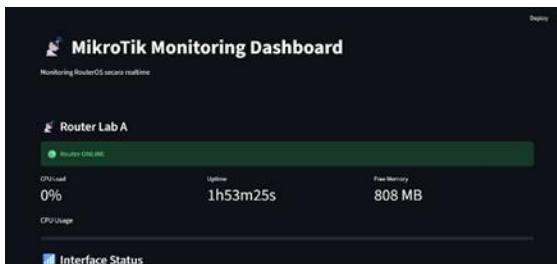


Fig. 11. Monitoring Dashboard in Online State



Fig. 12. Router Interface Status

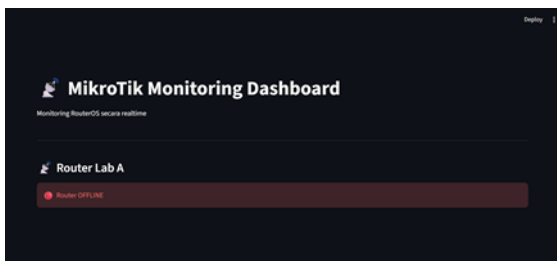


Fig. 13. Monitoring Dashboard in Offline State

E. System Testing

System testing was conducted by observing the response of the monitoring system under different router conditions. The first test was performed while the router was operating normally. The results showed that the system successfully detected the router status as ONLINE and displayed the corresponding device information on the Streamlit dashboard.

The second test simulated a network disruption scenario in which the router could not be accessed through the RouterOS API. The results demonstrated that the system successfully detected the OFFLINE condition, recorded the event in the monitoring log, and delivered automatic notifications through Telegram Bot. Furthermore, the Streamlit dashboard successfully updated the router status according to the latest network condition. These results indicate that the integration between RouterOS API, Telegram Bot, and Streamlit functions properly and reliably.

F. Discussion

The implementation results demonstrate that the proposed monitoring system is capable of monitoring router conditions in real time through the RouterOS API. The system not only detects network status changes but also provides additional device information, including CPU load, uptime, available memory, and network interface status.

The integration of Telegram Bot enables administrators to receive network status updates and disruption alerts quickly through mobile devices. Moreover, the availability of commands such as /status and /log increases system flexibility by allowing administrators to access monitoring information at any time.

From a visualization perspective, the Streamlit dashboard offers a more interactive monitoring environment compared to conventional terminal-based monitoring. Network conditions can be observed directly through a web-based interface without requiring manual inspection of the MikroTik device.

Compared with previous studies that primarily focused on notification mechanisms, the proposed system combines real-time monitoring, activity logging, Telegram notifications, and visual dashboard functionality within a single integrated platform. Therefore, the system can improve network monitoring efficiency and assist administrators in detecting and responding to network disruptions more effectively.

G. Research Limitation

This study has several limitations. First, the monitoring system was evaluated only within a virtualized laboratory environment and on a limited number of MikroTik routers. Second, the current implementation focuses on router availability monitoring and basic performance metrics such as CPU utilization, memory usage, uptime, and interface status. Third, the system does not yet provide advanced network analytics such as bandwidth monitoring, latency measurement, packet loss analysis, or predictive fault detection. These limitations may affect the applicability of the system in large-scale enterprise environments.

V. CONCLUSION

The study successfully implemented a Python-based MikroTik router monitoring system by integrating the RouterOS API, a Telegram Bot, and a Streamlit dashboard. The developed system is capable of performing real-time router monitoring, detecting both UP and DOWN status changes, recording monitoring activities in log files, and automatically sending notifications to network administrators through the Telegram Bot. In addition, the Streamlit dashboard effectively presents router information, including connection status, CPU utilization, memory capacity, uptime, and network interface status in an interactive and real-time manner.

The testing results demonstrate that the integration of the RouterOS API, Telegram Bot, and Streamlit dashboard functions effectively and reliably. The proposed system enhances the efficiency of network monitoring processes and

assists administrators in detecting and responding to network disruptions more quickly.

The proposed system has several limitations related to the evaluation environment, monitoring scope, and supported network performance metrics, as discussed in the previous section. For future work, the system can be extended to support monitoring of a larger number of network devices, provide historical network usage visualization through graphical analytics, and implement enhanced security mechanisms for communication between the monitoring application and network devices.

VI. REFERENCES

- [1] A. Yuazijah, A. Solehudin, and J. Haerul, "Sistem Monitoring Jaringan Menggunakan Metode Quality of Service (QoS) Dengan Perangkat Lunak The Dude," *Jurnal Mahasiswa Teknik Informatika*, vol. 8, no. 6, 2024, doi: 10.36040/jati.v8i6.11811.
- [2] T. Ariyadi and A. N. Riyansyah, "Penerapan Monitoring Jaringan dengan Zabbix pada PT. PLN (Persero) UIP Bagian Sumbagsel," *Jurnal Ilmiah Informatika (JIF)*, vol. 12, no. 1, 2024, doi: 10.33884/jif.v12i02.9283.
- [3] M. Saputra, "Desain dan Implementasi Monitoring Infrastruktur Perangkat Jaringan Menggunakan Grafana dan Prometheus," *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, vol. 12, no. 3, 2025, doi: 10.35957/jatisi.v12i3.13055.
- [4] M. Amin, M. Hasbi, B. Widada, and B. W. Yudanto, "Implementasi Monitoring Jaringan Internet pada Mikrotik Menggunakan Notifikasi Bot Telegram di SMK," *Jurnal Teknologi Informasi dan Komunikasi (TIKomsin)*, vol. 13, no. 1, p. 67, 2025, doi: 10.30646/tikomsin.v13i1.997.
- [5] M. Wahyu, A. S. Fitriani, and H. Hindarto, "Penerapan Bot Telegram untuk Sistem Monitoring Jaringan Intranet Daerah di Instansi Pemerintahan," *Infotek: Jurnal Informatika dan Teknologi*, vol. 7, no. 1, pp. 112–122, 2024, doi: 10.29408/jit.v7i1.24014.
- [6] T. Rahman, I. Z. Nibras, and S. Sumarna, "Monitoring Administrasi Jaringan dengan Mikrotik dan Telegram Bot pada Internet Service Provider," *Rabit: Jurnal Teknologi dan Sistem Informasi Univrab*, vol. 9, no. 2, pp. 162–172, 2024, doi: 10.36341/rabit.v9i2.4736.
- [7] M. Ichsan and F. Latifah, "Press-OK Monitoring Network Menggunakan Zabbix dengan Alert Notifikasi via Telegram pada PT XYZ," *Journal of Information System, Applied, Management, Accounting and Research (JISAMAR)*, vol. 9, no. 4, pp. 1340–1348, 2025, doi: 10.52362/jisamar.v9i4.2036.
- [8] F. P. E. Putra, K. Mufidah, R. M. Ilhamsyah, S. A. Efendy, and S. N. R. Barokah, "Tinjauan Performa RouterOS Mikrotik dalam Jaringan Internet: Analisis Kinerja dan Kelayakan," *Digital Transformation Technology*, vol. 3, no. 2, pp. 903–910, 2024, doi: 10.47709/digitech.v3i2.3446.
- [9] S. Kenat, "Analisis Keamanan Jaringan Menggunakan Mikrotik Pada Lab Komputer STMIK Widuri," *Neptunus: Jurnal Ilmu Komputer dan Teknologi Informasi*, vol. 2, no. 3, pp. 16–24, 2024, doi: 10.61132/neptunus.v2i3.177.
- [10] A. Dharmalau, G. Lienardy, N. Sucahyo, and I. Hiswara, "Implementasi Manajemen Bandwidth dan Firewall Menggunakan MikroTik Router pada Infrastruktur Jaringan di SMA Budi Mulia Jakarta," *JEIS: Jurnal Elektro dan Informatika Swadharma*, vol. 5, no. 1, pp. 19–27, 2025, doi: 10.56486/jeis.vol5no1.514.
- [11] S. Nugroho and B. Pujiarto, "Network Automation Pada Beberapa Perangkat Router Menggunakan Pemrograman Python," *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK)*, vol. 9, no. 1, pp. 79–86, 2022, doi: 10.25126/jtik.202293947.
- [12] A. Kurniawan et al., "Penerapan Network Automation Pada Server Menggunakan Python Berbasis Web," *SNIV: Seminar Nasional Inovasi Vokasi*, vol. 4, no. 1, 2025.
- [13] H. Prasetyo et al., "Penerapan Network Automation pada Lingkungan Jaringan Berbasis Web," *Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, vol. 4, no. 2, 2025.
- [14] F. Amalia Maresti, M. Zaidee Aulia, P. Bota Kolin, and K. Mustaqim, "Implementasi Framework Streamlit Untuk Menentukan Besaran Dana Bantuan Berdasarkan Prioritas Dengan Metode Weighted Product," *DATA SCIENCES INDONESIA*, vol. 6, no. 1, 2026, doi: 10.47709/dsi.v6i1.7849.
- [15] K. Nabila Oktaviarini, E. D. Wahyuni, and R. Permata Sari, "Transformasi Data Statistik Menjadi Visual Interaktif Menggunakan Streamlit: Studi Kasus BPS Kota Mojokerto," *Jurnal Kridatama Sains dan Teknologi*, vol. 6, no. 2, 2024, doi: 10.53863/kst.v6i02.1426.
- [16] F. Jamil Al Fajar, D. Nurani, and R. Faticha Alfa Aziza, "Implementasi Bot Telegram untuk Monitoring Mikrotik pada Planets Network Solution," *Jurnal Teknologi dan Komputer*, vol. 9, no. 6, 2023, doi: 10.36002/jutik.v9i6.2670.
- [17] R. Hartono and M. N. Kholidah, "Sistem User Management Menggunakan Mikrotik RouterOS untuk Mengontrol Akses Internet Siswa pada SMK Miftakhul Huda Way Areng," *Jurnal Ilmiah Sistem Informasi dan Ilmu Komputer*, vol. 4, no. 2, pp. 18–28, 2024, doi: 10.55606/juisik.v4i2.1828.
- [18] A. A. Pertiwi et al., "Implementasi Sistem Logging Jaringan Berbasis Web untuk Monitoring dan Analisis Real-Time," *Jurnal Informatika dan Rekayasa Komputer (JAKAKOM)*, vol. 6, no. 1, 2026.
- [19] A. Arizki and N. A. Chandra, "Optimalisasi Jaringan Internet Dengan Router Mikrotik Menggunakan Teknik Load Balancing Memakai Metode PCC (Per Connection Classifier) Dengan Topologi Hybrid," *Journal of Computing Engineering, System and Science*, vol. 9, no. 2, 2024.
- [20] M. R. A. Arsandi and A. Syaripudin, "Perancangan Sistem Monitoring Jaringan Berbasis Web Server Terintegrasi Zabbix dan Notifikasi Telegram pada PT Time Excelindo," *OKTAL: Jurnal Ilmu Komputer dan Sains*, vol. 3, no. 6, pp. 1553–1561, 2024.

This page is intentionally left blank