Project 5

Web Interface for Network Management

ITCS465 NETWORK MANAGEMENT

NAME:		ID:
Sirapath	Thainiyom	6488108
Suphavadee	Cheng	6488120
Jidapa	Moolkaew	6488176

01 Introduction

Web Interface for Network Management

- To create a centralized web interface for monitoring and managing network traffic.
- Display metrics such as bandwidth usage, incoming/outgoing traffic, and provide historical data for analysis.

> This project is focused on creating a web interface for network management that allows users to monitor, analyze, and manage the network's performance in real time.

02 Core Features

Real-Time Data Visualization

- View live bandwidth usage and network performance.
- Line and curve charts for bandwidth in/out metrics.
- Customizable time range filters (hour, day, week).

Network Metrics Dashboard

- Displays key statistics (current, average, max, and min values for network bandwidth).

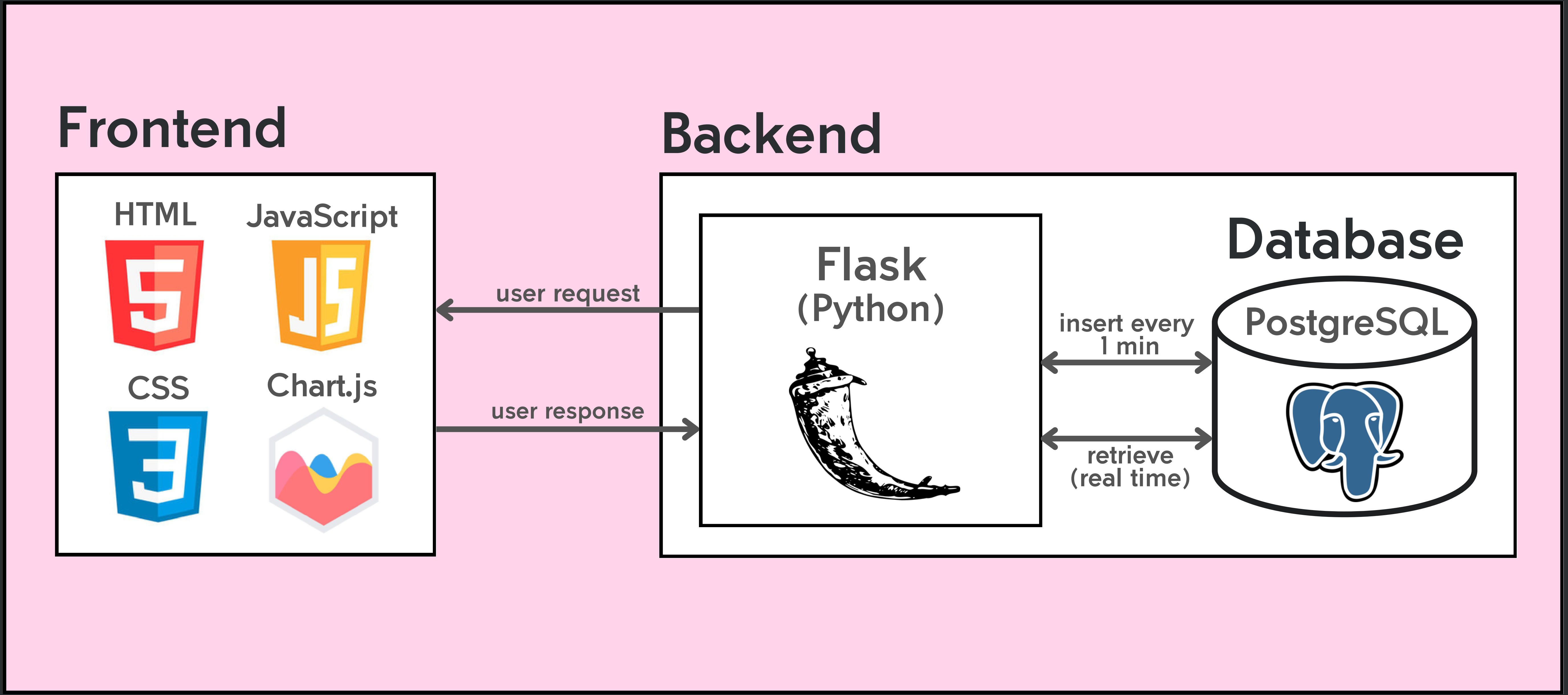
Device Management

- Easily integrate network devices for monitoring using SNMP.
- Accessible from any device with a browser.

Interactive User Interface

- Easy-to-navigate design with responsive layouts.
- Responsive design with smooth navigation for better user experience.

03 System Architecture



04 Implementation Tools

Frontend

- User interface built with **HTML, CSS, JavaScript,** and **Chart.js** for visualizations.

Backend

- Handles data processing and API communication using **Python (Flask)**
- **SQLAlchemy**
- **pysnmp**
- **psycopg2**

Database

PostgreSQL stores the historical and real-time network data

Schema

Table Name: SNMPCriticalMetrics

id	Unique identifier
metric_name	Name of the metric
oid	SNMP Object Identifier for the metric
value	Raw SNMP value
value_type	Type of value
ip_port	IP address and port of the monitored device
timestamp	When the metric was collected

Database (1)

OID Definition for Metrics

Critical Metrics used in The System

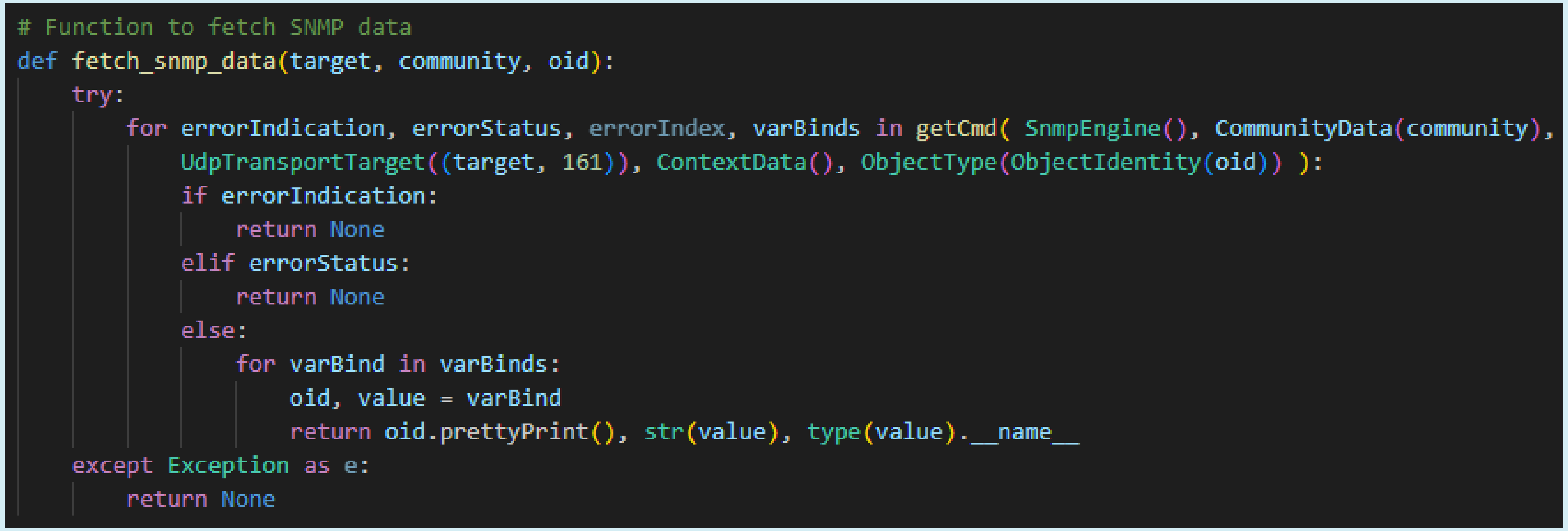
Bandwidth In	:	1.3.6.1.2.1.2.2.1.10
Bandwidth Out	:	1.3.6.1.2.1.2.2.1.16
Input Errors	:	1.3.6.1.2.1.2.2.1.14
Output Errors	:	1.3.6.1.2.1.2.2.1.20
System Uptime	:	1.3.6.1.2.1.1.3.0
IP Packets Received	:	1.3.6.1.2.1.4.3.0
UDP Datagrams Sent	:	1.3.6.1.2.1.7.4.0
TCP Connections	:	1.3.6.1.2.1.6.9.0
Incoming IP Errors	:	1.3.6.1.2.1.4.5.0

- OIDs (Object Identifiers) are used to identify SNMP metrics from network devices.

Database (2)

Fetching SNMP Data

- It retrieves the value of specific OIDs (Object Identifiers), which represent various network metrics.
- The SNMP data is fetched **every minute** (set by time.sleep(60)), processed, and inserted into the PostgreSQL database.

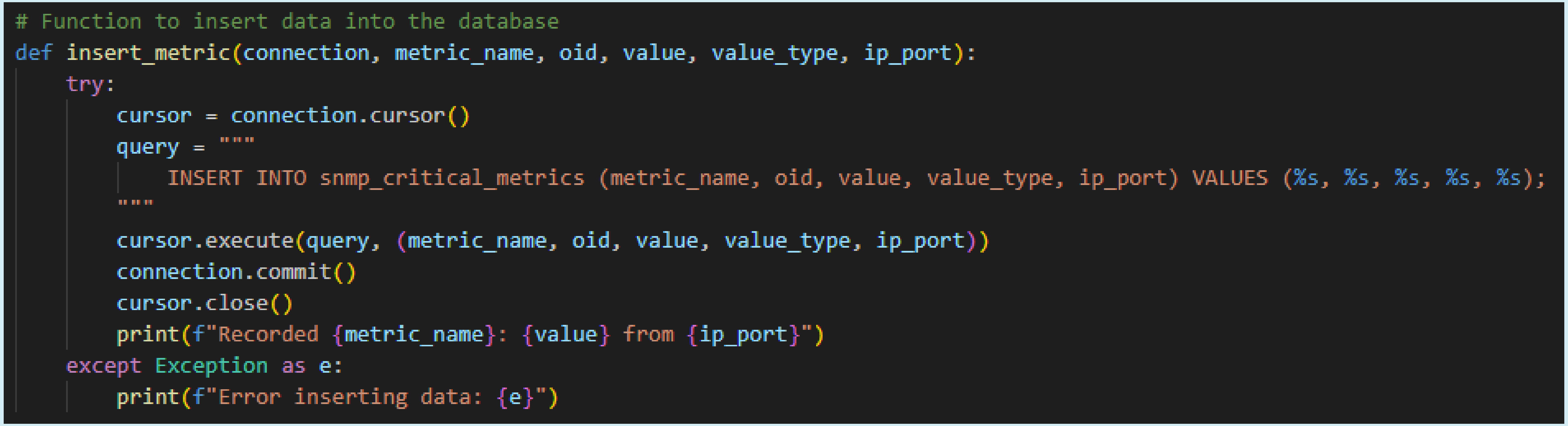


```python
# Function to fetch SNMP data
def fetch_snmp_data(target, community, oid):
    try:
        for errorIndication, errorStatus, errorIndex, varBinds in getCmd( SnmpEngine(), CommunityData(community),
            UdpTransportTarget((target, 161)), ContextData(), ObjectType(ObjectIdentity(oid)) ):
            if errorIndication:
                return None
            elif errorStatus:
                return None
            else:
                for varBind in varBinds:
                    oid, value = varBind
                    return oid.prettyPrint(), str(value), type(value).__name__
    except Exception as e:
        return None
```

fetch_snmp_data(): This function is responsible for collecting data from network devices using the SNMP protocol.

Database (3)

Data Collection & Insertion into Database

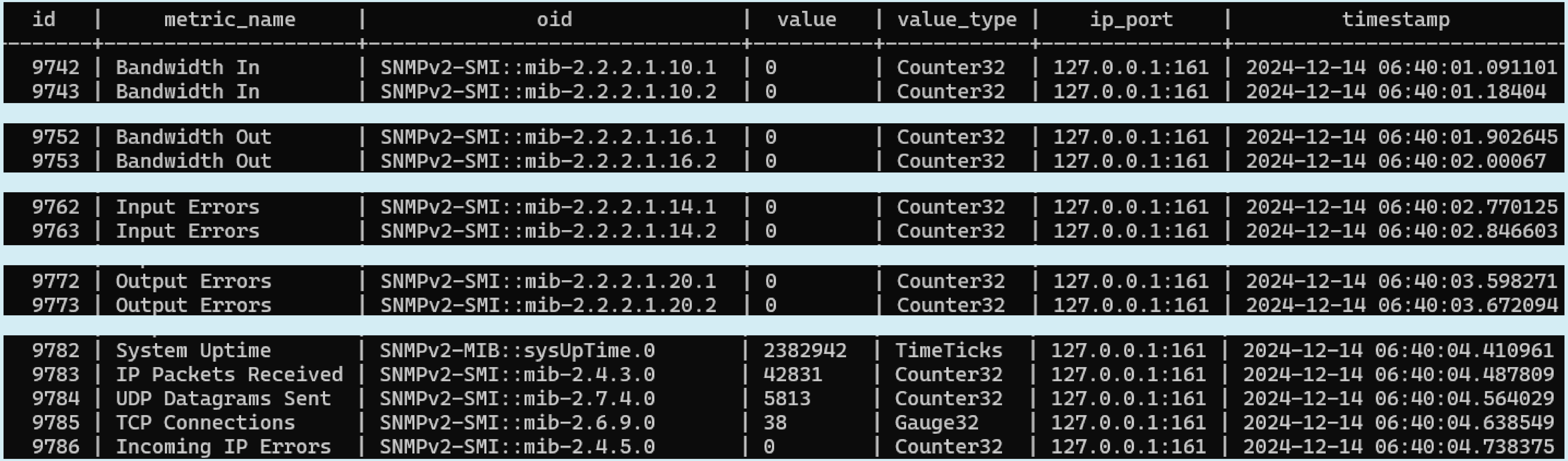


```python
# Function to insert data into the database
def insert_metric(connection, metric_name, oid, value, value_type, ip_port):
    try:
        cursor = connection.cursor()
        query = """
            INSERT INTO snmp_critical_metrics (metric_name, oid, value, value_type, ip_port) VALUES (%s, %s, %s, %s, %s);
        """
        cursor.execute(query, (metric_name, oid, value, value_type, ip_port))
        connection.commit()
        cursor.close()
        print(f"Recorded {metric_name}: {value} from {ip_port}")
    except Exception as e:
        print(f"Error inserting data: {e}")
```

insert_metric(): This function inserts the SNMP data into the snmp_critical_metrics table in PostgreSQL.

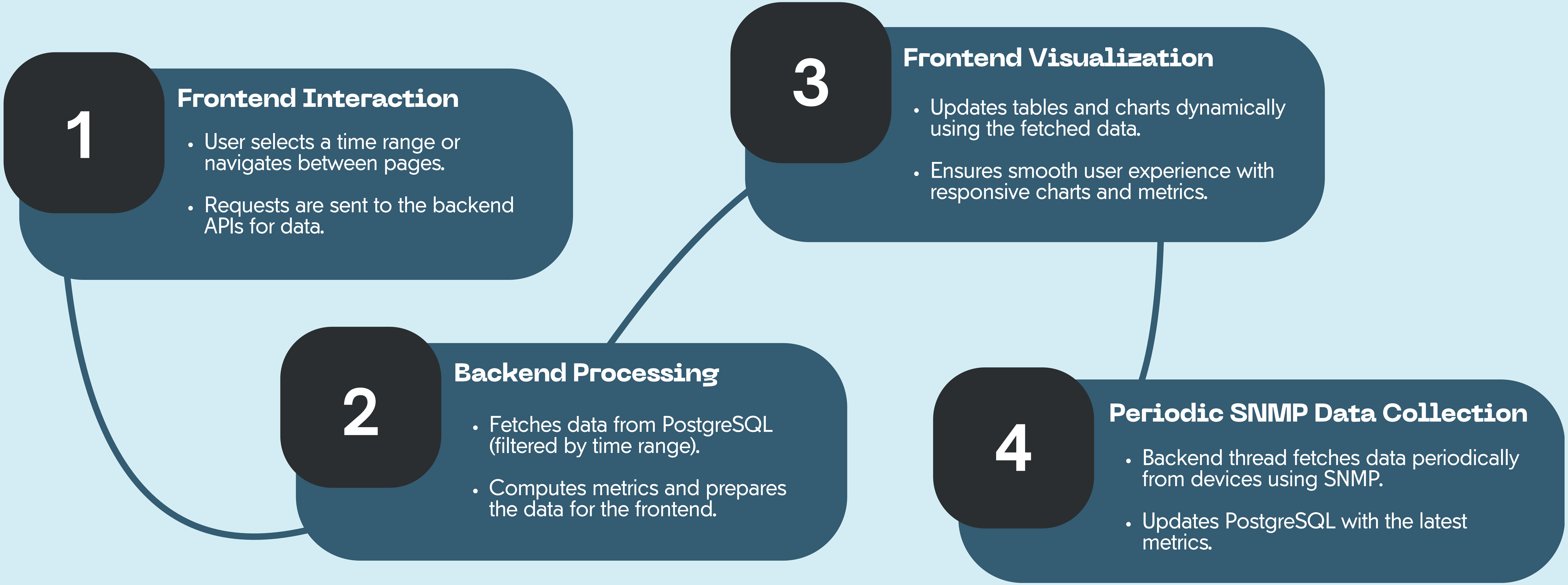
Database (4)

Fetching Data Example



```
  id   |     metric_name      |               oid                |  value   | value_type |    ip_port    |          timestamp
-------+----------------------+----------------------------------+----------+------------+---------------+----------------------------
  9742 | Bandwidth In         | SNMPv2-SMI::mib-2.2.2.1.10.1     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:01.091101
  9743 | Bandwidth In         | SNMPv2-SMI::mib-2.2.2.1.10.2     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:01.18404

  9752 | Bandwidth Out        | SNMPv2-SMI::mib-2.2.2.1.16.1     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:01.902645
  9753 | Bandwidth Out        | SNMPv2-SMI::mib-2.2.2.1.16.2     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:02.00067

  9762 | Input Errors         | SNMPv2-SMI::mib-2.2.2.1.14.1     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:02.770125
  9763 | Input Errors         | SNMPv2-SMI::mib-2.2.2.1.14.2     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:02.846603

  9772 | Output Errors        | SNMPv2-SMI::mib-2.2.2.1.20.1     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:03.598271
  9773 | Output Errors        | SNMPv2-SMI::mib-2.2.2.1.20.2     | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:03.672094

  9782 | System Uptime        | SNMPv2-MIB::sysUpTime.0          | 2382942  | TimeTicks  | 127.0.0.1:161 | 2024-12-14 06:40:04.410961
  9783 | IP Packets Received  | SNMPv2-SMI::mib-2.4.3.0          | 42831    | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:04.487809
  9784 | UDP Datagrams Sent   | SNMPv2-SMI::mib-2.7.4.0          | 5813     | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:04.564029
  9785 | TCP Connections      | SNMPv2-SMI::mib-2.6.9.0          | 38       | Gauge32    | 127.0.0.1:161 | 2024-12-14 06:40:04.638549
  9786 | Incoming IP Errors   | SNMPv2-SMI::mib-2.4.5.0          | 0        | Counter32  | 127.0.0.1:161 | 2024-12-14 06:40:04.738375
```

05 Workflow



1 Frontend Interaction

- User selects a time range or navigates between pages.
- Requests are sent to the backend APIs for data.

2 Backend Processing

- Fetches data from PostgreSQL (filtered by time range).
- Computes metrics and prepares the data for the frontend.

3 Frontend Visualization

- Updates tables and charts dynamically using the fetched data.
- Ensures smooth user experience with responsive charts and metrics.

4 Periodic SNMP Data Collection

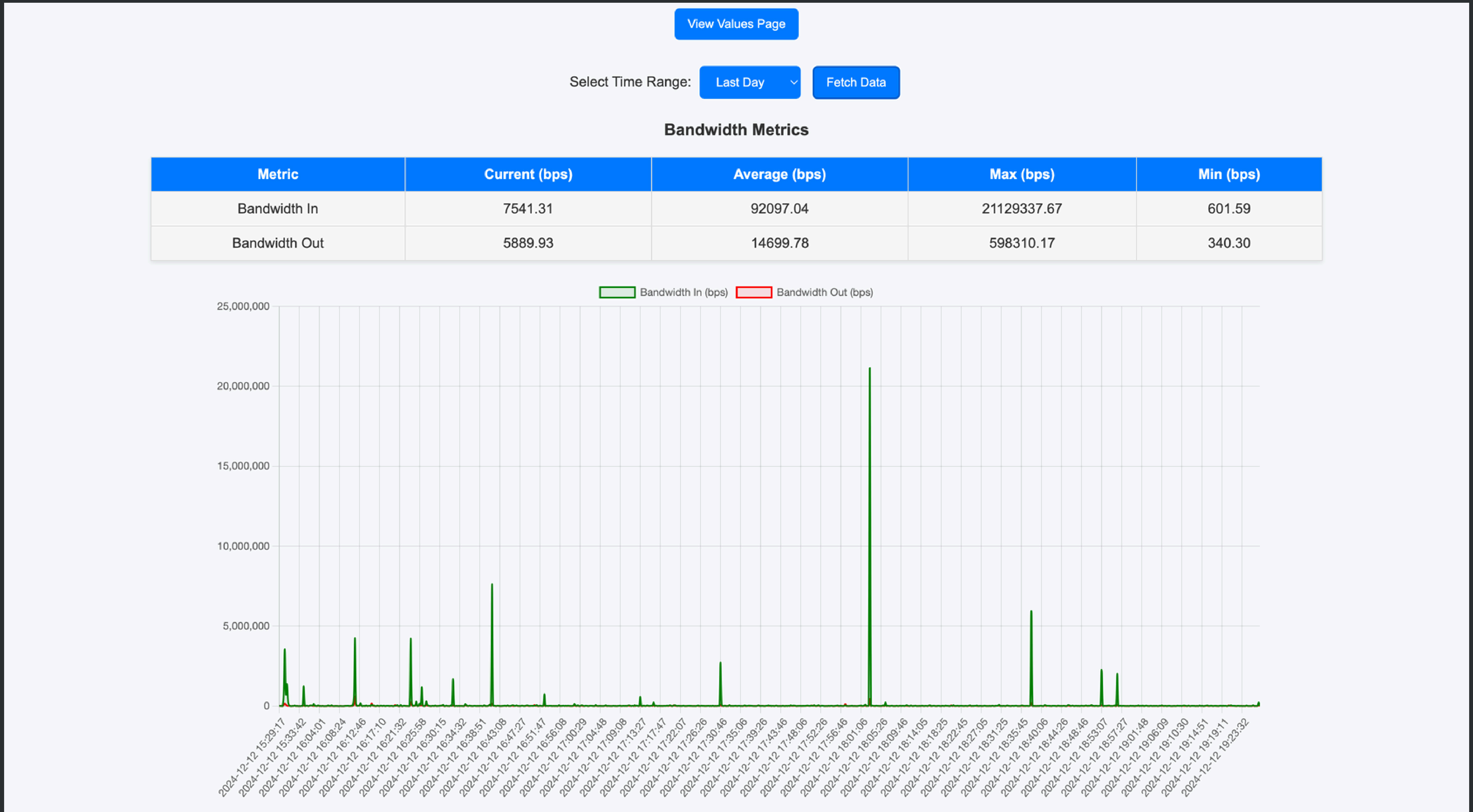
- Backend thread fetches data periodically from devices using SNMP.
- Updates PostgreSQL with the latest metrics.

06

Web Interface

06 Web Interface

Bandwidth Traffic Analysis (Rate) Page

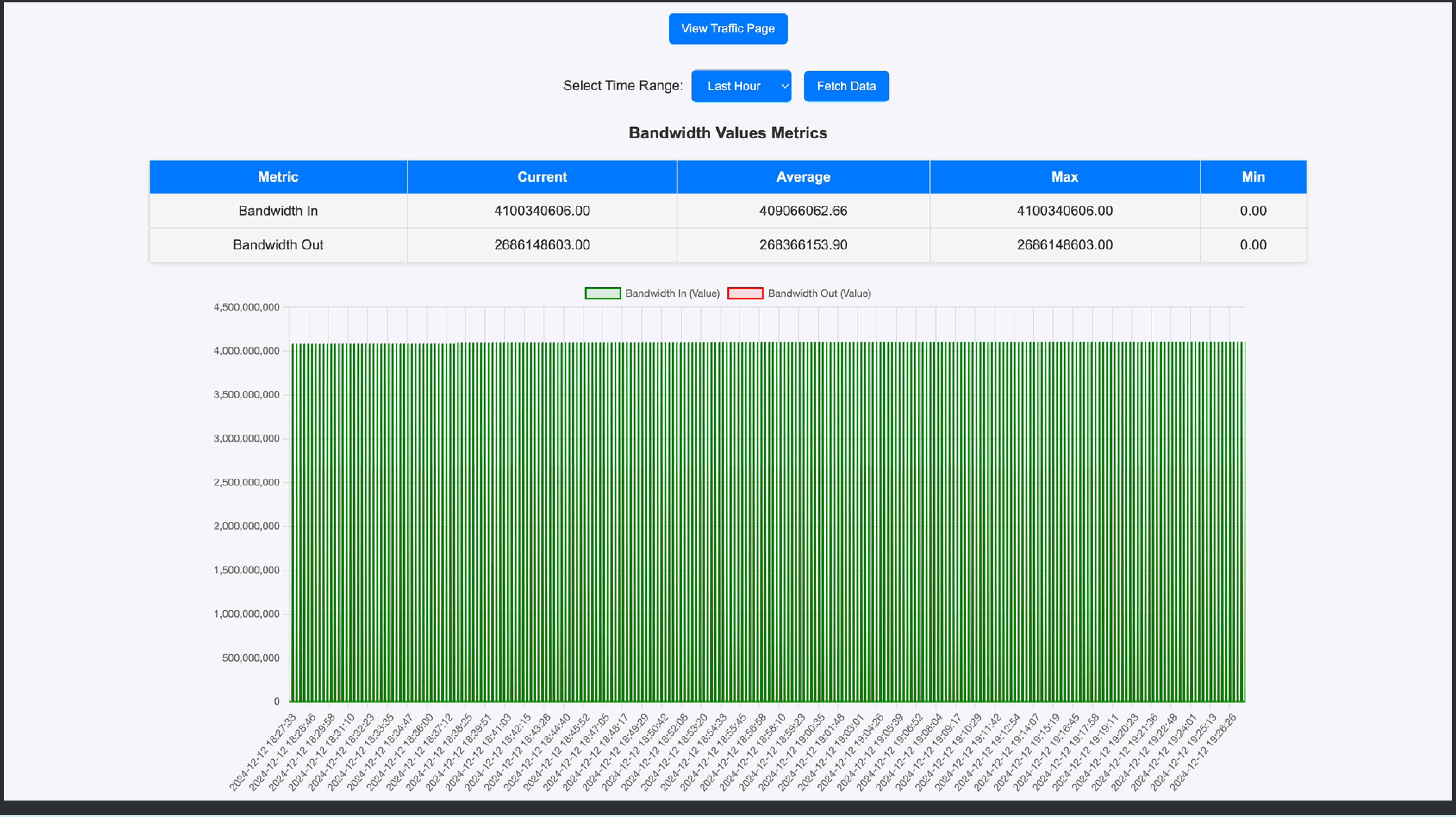


Bandwidth Metrics

Metric	Current (bps)	Average (bps)	Max (bps)	Min (bps)
Bandwidth In	7541.31	92097.04	21129337.67	601.59
Bandwidth Out	5889.93	14699.78	598310.17	340.30

06 Web Interface

Bandwidth Traffic Value Page



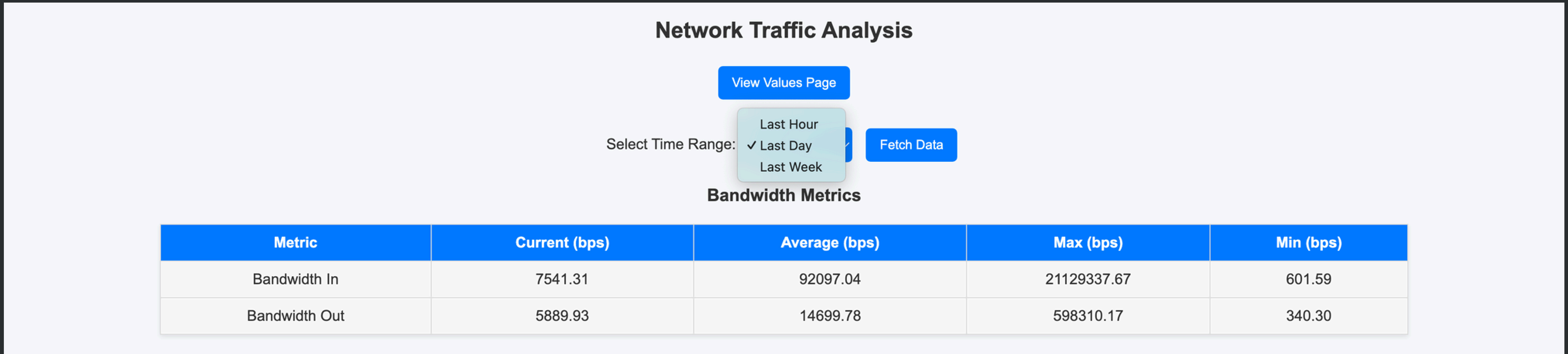
Select Time Range: Last Hour Fetch Data

Bandwidth Values Metrics

Metric	Current	Average	Max	Min
Bandwidth In	4100340606.00	409066062.66	4100340606.00	0.00
Bandwidth Out	2686148603.00	268366153.90	2686148603.00	0.00

06 Web Interface

Traffic Matrics Queries Feature

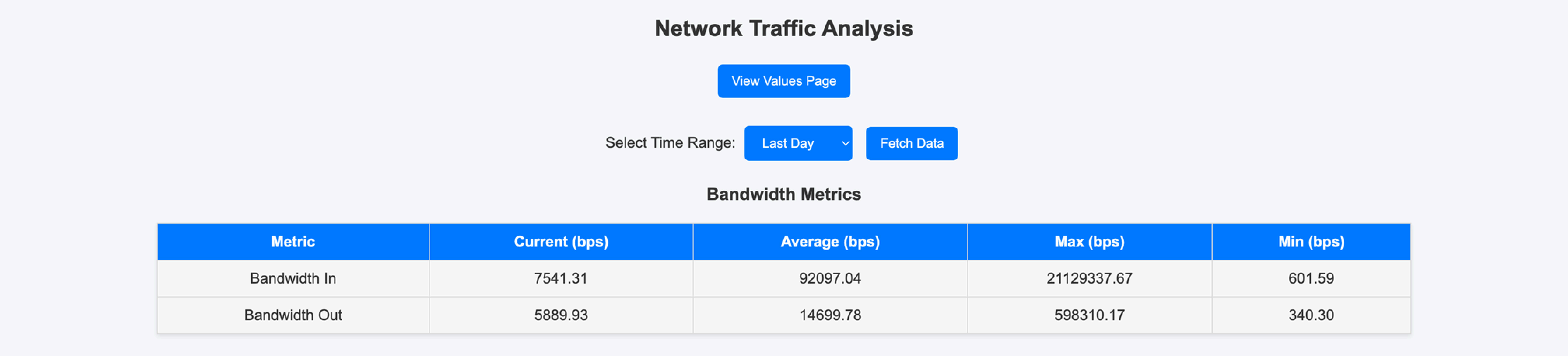


Network Traffic Analysis

Select Time Range: Last Day

Bandwidth Metrics

Metric	Current (bps)	Average (bps)	Max (bps)	Min (bps)
Bandwidth In	7541.31	92097.04	21129337.67	601.59
Bandwidth Out	5889.93	14699.78	598310.17	340.30



Network Traffic Analysis

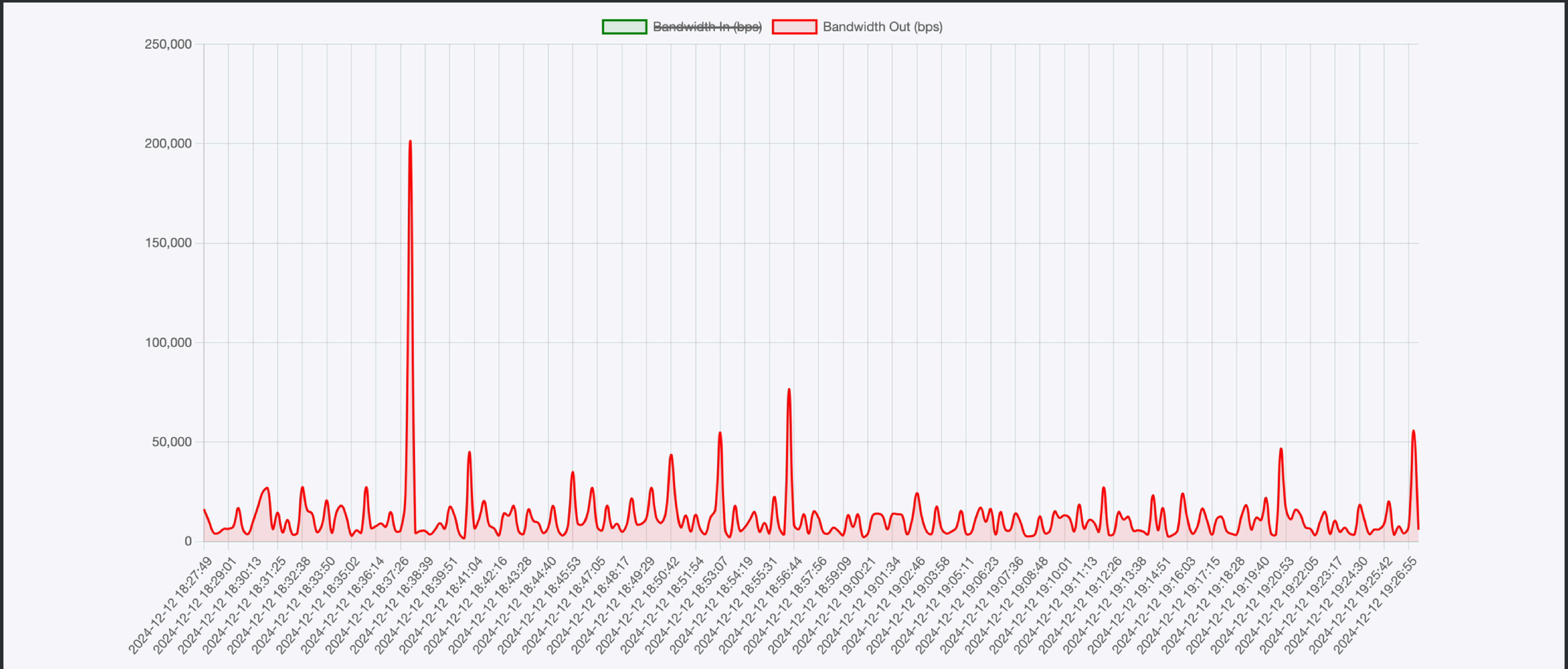
Select Time Range: Last Day

Bandwidth Metrics

Metric	Current (bps)	Average (bps)	Max (bps)	Min (bps)
Bandwidth In	7541.31	92097.04	21129337.67	601.59
Bandwidth Out	5889.93	14699.78	598310.17	340.30

06 Web Interface

Graph Display Selection



07 Challenges Faced and Solutions

Challenge	Solution
Accurate Rate Calculation Handling SNMP counter resets and ensuring accurate rate calculations for Bandwidth In and Bandwidth Out.	Implemented logic to detect counter resets and handle them by resetting value_diff to the current value.
SNMP Data Retrieval Not Working When attempting to retrieve SNMP data for the first time, common issues include SNMP not being enabled on the target device or incorrect configuration.	Ensure SNMP is enabled, the correct community string is used, and there are no firewall restrictions blocking UDP port 161. Once configured, you can successfully retrieve data using SNMP tools like snmpwalk.

08 Future Enhancements

Threshold-Based

Implement a system for defining thresholds for metrics (e.g., high bandwidth usage).

Additional Metrics

Include total data transferred (total bytes sent and received) alongside current rates and add metrics like total active connections, packet loss rates, and latency.

Virtual Interface Monitoring

Future updates will add support for monitoring multiple virtual interfaces and dynamic graphs.

Improved Database Efficiency

The database will store additional metrics like data transfer, active connections, and latency for better performance tracking.

09 Conclusion

This project successfully implements a comprehensive network monitoring solution using SNMP, Flask, and PostgreSQL. Key features include real-time traffic analysis, device summary monitoring, and dynamic visualizations.

Thank You.