Test Plan for Bandwidth Monitoring System

Version: 1.0
Prepared by: B. Jathin Reddy

1.1 Objectives

- Verify accurate data collection (bytes sent/received).
- Validate alerting logic (threshold breaches).
- Ensure dashboard visualizations reflect real-time/historical data.

1.2 Scope

- **In Scope:**
    - API endpoints (/api/current_usage, /api/alerts).
    - Data logging (psutil → SQLite).
    - Dashboard UI (dashboard.html).
- **Out of Scope:**
    - Cross-browser testing (assume Chrome/Firefox).
    - Load testing (single-user focus).

---

2. Test Scenarios

Scenario 1: Real-Time Data Collection & Display

Objective: Verify the system correctly logs and displays live bandwidth usage.

Test Steps:

1. **Setup:**
    - Start the Flask server (python app.py).
    - Open dashboard.html in a browser.
2. **Action:**
    - Generate network traffic (e.g., download a file, stream video).
3. **Validation:**
    - Check if Bytes Sent/Received values update every 5s (matching UPDATE_INTERVAL).
    - Confirm charts (usage-chart) reflect real-time spikes.
    - Verify Last updated timestamp changes dynamically.

Expected Result:

- Dashboard shows live updates with correct byte counts.
- Charts plot data without delays.

Scenario 2: Alert Generation for High Bandwidth Usage

Objective: Ensure alerts trigger when usage exceeds 100 MB.

Test Steps:

1. **Setup:**
    - Set ALERT_THRESHOLD = 100 * 1024 * 1024 (100 MB) in app.py.
    - Purge old alerts: DELETE FROM alerts; in SQLite.
2. **Action:**
    - Simulate heavy traffic (e.g., run dd if=/dev/zero bs=1M count=200 | nc localhost 8000).
3. **Validation:**
    - Check the Alerts tab in the dashboard for a new alert.
    - Query SQLite: SELECT * FROM alerts WHERE resolved = 0;.

Expected Result:

- Alert appears on screen
- Database logs the alert with actual_value > threshold.

Scenario 3: Historical Data Retrieval & Visualization

Objective: Validate time-range filters (1h/24h/7d) for historical data.

Test Steps:

1. **Setup:**
    - Ensure bandwidth_usage table has ≥7 days of test data.
2. **Action:**
    - Switch between tabs in the dashboard:
        - *1 Hour* → Verify hourly granularity.
        - *24 Hours* → Check daily aggregates.
        - *7 Days* → Confirm weekly trends.
3. **Validation:**
    - Inspect API responses (/api/historical?range=24h).

- Ensure charts resize/rebuild correctly.

Expected Result:

- Charts show correct time ranges without missing data.
- API returns non-empty times, sent, received arrays.