# Software Design Document

## NetworkWeaver: A Centralized Network Control and Monitoring Platform Using SNMP and APIs

**NetworkWeaver Team:**

# Table of Contents


1. **Introduction**
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions, Acronyms and Abbreviations
    - 1.4 References
2. **System Overview**
3. **System Components**
    - 3.1 Decomposition Description
    - 3.2 Dependency Description
    - 3.3 Interface Description
    - 3.4 Module Interfaces
    - 3.5 User Interfaces (GUI)
4. **Detailed Design**
    - 4.1 Module Detailed Design
    - 4.2 Data Detailed Design
    - 4.3 Requirements Traceability Matrix (RTM)

# 1. Introduction

## 1.1 Purpose

The purpose of this Software Design Document (SDD) is to describe the architecture and system design for **NetworkWeaver**, a centralized network monitoring and configuration web application. This document serves as a guide for the development team to implement the system using a local deployment strategy and ensures all functional requirements regarding MikroTik device management are met.

## 1.2 Scope

This document describes the implementation details of the NetworkWeaver Web Application. NetworkWeaver addresses the complexity of managing distributed network infrastructure by unifying monitoring and configuration into a single platform.

- **Monitoring Module:** Integration of Prometheus and Grafana to visualize real-time network data (CPU, Memory, Traffic) collected via SNMP Exporter.
- **Configuration Module:** A Python-based automation engine that uses the MikroTik RouterOS v7 API to deploy configuration templates and scripts.
- **Local Deployment:** The system is containerized using Docker and hosted locally to ensure data privacy and reduce reliance on cloud connectivity, utilizing a local PostgreSQL database instead of DynamoDB.
- **Security:** Implementation of WireGuard VPN to secure communication between the system and managed devices

## 1.3 Definitions, Acronyms and Abbreviations

| Acronym | Meaning |
|---|---|
| **SDD** | Software Design Document |
| **SNMP** | Simple Network Management Protocol |
| **API** | Application Programming Interface |
| **GUI** | Graphical User Interface |
| **VPN** | Virtual Private Network (WireGuard) |
| **OS** | Operating System (RouterOS) |
| **SME** | Small-to-Medium Enterprise |

### 1.4 References


- Arenas, M.J., et al. "Network Weaver: A Centralized Network Control and Monitoring Platform Using SNMP and APIs" (Thesis Paper).
- MikroTik RouterOS Documentation (v7).
- Prometheus & Grafana Official Documentation.


# 2. System Overview

**Architectural Pattern:** Microservices (Containerized)

NetworkWeaver utilizes a modular architecture deployed via **Docker** containers. This ensures consistency across development and testing environments (Virtual Labs like GNS3/EVE-NG).

- **Frontend:** A **React.js** web interface that serves as the dashboard for administrators. It embeds Grafana panels for monitoring and provides forms for configuration inputs.

- **Backend API:** A **Python (FastAPI)** application that acts as the orchestrator. It handles user requests, interacts with the Database, and executes configuration scripts against routers.

- **Monitoring Stack:** Composed of **SNMP Exporter** (data collection), **Prometheus** (time-series database), and **Grafana** (visualization).

- **Database:** A local **PostgreSQL** database stores user credentials, device inventory, and configuration logs.

- **Connectivity: WireGuard** establishes secure tunnels between the Docker host and the MikroTik routers.

*(Insert High-Level Architecture Diagram here) Note: The diagram should show the React Frontend connecting to the FastAPI Backend. Parallel to this, the SNMP Exporter polls Routers, sends data to Prometheus, which feeds Grafana. Grafana dashboards are then embedded back into the React Frontend.)*

# 3. System Components

### 3.1 Decomposition Description

**Top-Down Details:**

1. **UI Controller (Frontend):**
    - Manages the React.js components.
    - Handles "Monitoring View" (embedding Grafana iframe) and "Config View" (forms for scripts).
2. **API Gateway (FastAPI Backend):**
    - **Device Manager:** Handles adding/removing routers (IP, Credentials).
    - **Config Executor:** Receives script parameters from UI, formats them, and pushes them to MikroTik via RouterOS API.
    - **Auth Service:** Validates user login sessions.
3. **Monitoring Subsystem:**
    - **SNMP Exporter:** Translates SNMP OIDs from MikroTik into Prometheus metrics.
    - **Prometheus:** Scrapes metrics at set intervals (e.g., 15s).
    - **Grafana:** Queries Prometheus to build charts (Bandwidth, CPU Load).
4. **Data Persistence:**
    - **PostgreSQL:** Stores relational data (Users, Devices, Logs).

*(Insert Component Decomposition Diagram here)*

### 3.2 Dependency Description

- **Frontend** depends on **Backend API** for data and **Grafana** for visuals.
- **Backend API** depends on **PostgreSQL** for storage and **RouterOS API** for device control.
- **Prometheus** depends on **SNMP Exporter** to fetch raw data.
- **Grafana** depends on **Prometheus** as its data source.

### 3.3 Interface Description

#### 3.3.1 Frontend to Backend API Interface

- **Protocol:** HTTP/JSON
- **Functions:**
    - login(username, password)
    - getDevices()
    - deployConfig(deviceId, scriptId, params)

### 3.3.2 Backend to RouterOS Interface

- **Protocol:** RouterOS API (v7) / TCP
- **Functions:**
    - /ip/address/add (Example command execution)
    - /system/reboot

### 3.3.3 Monitoring Interface

- **Protocol:** SNMP (UDP 161)
- **Flow:** MikroTik -> SNMP Exporter -> Prometheus -> Grafana.

### 3.4 Module Interfaces

*(Insert Diagram showing data flow between React, FastAPI, Postgres, and the Monitoring Stack)*

### 3.5 User Interfaces (GUI)

**Landing Page:**

- Login screen requiring Username and Password.

**Dashboard (Home):**

- Displays a list of active MikroTik devices.
- Shows aggregate network health (Online/Offline status).
- *(Insert Mockup of Dashboard here)*

**Monitoring Tab:**

- Embedded Grafana panels showing:
    - Interface Traffic (Mbps)
    - CPU & Memory Usage
- *(Insert Mockup of Monitoring View here)*

**Configuration Tab:**

- **Template Selector:** Dropdown to choose scripts (e.g., "Basic Firewall Setup", "Bandwidth Limit").
- **Parameter Inputs:** Fields to enter IP addresses or limit values.
- **Action Buttons:** "Deploy", "Rollback".
- *(Insert Mockup of Configuration Form here)*

# 4. Detailed Design

## 4.1 Module Detailed Design

### 4.1.1 Configuration Automation Module

**Sequence Diagram:**

1. **User** selects "Block Website" template on **Frontend**.
2. **Frontend** sends POST request to **FastAPI** with target URL and Router ID.
3. **FastAPI** retrieves Router credentials from **PostgreSQL**.
4. **FastAPI** connects to **MikroTik Router** via RouterOS API.
5. **FastAPI** executes the Python script to add a firewall layer 7 protocol rule.
6. **Router** confirms execution.
7. **FastAPI** logs the action in **PostgreSQL** and returns "Success" to **Frontend**.

*(Insert Sequence Diagram here)*

Pseudocode (Python/FastAPI):

```
def deploy_configuration(device_id, script_template, params):
    # Fetch device details
    device = database.get_device(device_id)

    # Connect via RouterOS API
    connection = routeros_api.connect(device.ip, device.user, device.password)

    # Prepare command based on template
    command = script_template.format(params)

    # Execute
    try:
        connection.run(command)
        database.log_action(device_id, "Config Applied", "Success")
        return "Configuration Successful"
   except Exception as e:
        database.log_action(device_id, "Config Failed", str(e))
        return "Error: " + str(e)
```

### 4.1.2 Monitoring Module (SNMP Integration)

- **Mechanism:** The SNMP Exporter is configured with a snmp.yml file mapping MikroTik MIBs (Management Information Bases) to readable metrics.
- **Data Flow:** Prometheus scrapes the exporter endpoint http://snmp-exporter:9116/metrics every 15 seconds.

## 4.2 Data Detailed Design (PostgreSQL Schema)

### Table: Users

`id` (PK), `username`, `password_hash`, `role`

### Table: Devices

`id` (PK), `name`, `ip_address`, `vpn_ip`, `api_port` (default 8728), `snmp_community`

### Table: Configuration_Logs

`log_id` (PK), `device_id` (FK), `timestamp`, `action_type`, `status`, `details`

## 4.3 Requirements Traceability Matrix (RTM)

| Requirement ID | Requirement Description | Design Component | Test Case ID |
|---|---|---|---|
| **REQ-01** | System must collect real-time traffic data via SNMP. | Monitoring Subsystem (Prometheus/SNMP Exporter) | TC-MON-01 |
| **REQ-02** | System must allow remote configuration of RouterOS v7. | Backend API (Python/RouterOS Lib) | TC-CFG-01 |
| **REQ-03** | System must secure device communication. | WireGuard VPN Container | TC-SEC-01 |
| **REQ-04** | User must be able to view logs of changes. | PostgreSQL Database & Frontend History View | TC-LOG-01 |
| **REQ-05** | System must be deployed locally without cloud dependency. | Docker Compose | TC-DEP-01 |