OPERATING SYSTEMS

**PROJECT**

# Section: K23SA

*Faculty:* ***Dr. Gurbinder Singh Brar***

*Name:* ***Amit Tiwari***

*Registration No.:* ***12304569***

# *TOPIC:* Real-Time Process Monitoring Dashboard

# *Repository name:* Resource\_Monitoring\_System

# *Repository link:* <https://github.com/AmiTtiwari43/Resource_Monitoring_System.git>

****

*Real-Time Process Monitoring Dashboard*

# REAL-TIME PROCESS MONITORING DASHBOARD

The **Real-Time Process Monitoring Dashboard** is an advanced graphical interface designed to provide administrators with real-time insights into the performance and status of system processes. By continuously monitoring key metrics such as **process states, CPU usage, and memory consumption**, this dashboard enables proactive system management, helping to swiftly detect and resolve potential performance issues.

# Key Features

1. **Real-Time System Insights**
   * Displays active processes and their states, ensuring administrators have an up-to-date view of system activity.
   * Continuously tracks CPU and memory utilization, allowing for efficient resource management.

# Intuitive Graphical Interface

* + Provides interactive charts and visual indicators that simplify performance analysis.
  + Offers a customizable layout for administrators to focus on specific metrics based on their needs.

# Performance Analysis & Alerts

* + Detects unusual spikes or anomalies in resource consumption, helping to identify potential system slowdowns or failures.
  + Sends real-time alerts when predefined thresholds are breached, ensuring timely intervention.

# Historical Data & Trend Analysis

* + Maintains logs of past performance metrics to support detailed trend analysis.
  + Helps in forecasting potential system bottlenecks based on historical patterns.

# Process Control & Management

* + Allows administrators to pause, resume, or terminate processes directly from the dashboard.
  + Provides insights into resource-hungry applications, helping optimize system efficiency.

# Benefits

* **Enhanced System Performance**: Helps administrators maintain an optimized and well-managed system by tracking resource usage.
* **Proactive Issue Detection**: Enables rapid troubleshooting and problem resolution through real-time monitoring and alerts.
* **Improved Decision-Making**: Provides actionable insights based on historical data and trends, assisting in resource planning and system improvements.
* **Efficient Resource Allocation**: Ensures balanced CPU and memory usage to prevent system overload or inefficiency.

# Conclusion

The **Real-Time Process Monitoring Dashboard** is an essential tool for administrators aiming to maintain a high-performance computing environment. By offering real-time insights, alerting mechanisms, and historical trend analysis, it significantly enhances system management and optimizes overall efficiency.

Would you like me to refine specific sections or add additional details?

# Table of Contents

1. Introduction
2. Objectives
3. Technologies Used
4. System Design
5. Implementation
6. Features
7. Conclusion
8. Future Enhancements

# Real-Time Process Monitoring Dashboard: Detailed Explanation

1. **Introduction**

Efficient system monitoring plays a vital role in ensuring the **optimal performance and stability** of computing environments. The **Real-Time Process Monitoring Dashboard** is a **web-based application** designed to provide administrators with **real-time insights** into system resource usage, including **CPU, memory, disk, and network activity**.

This dashboard offers **a visual representation** of system performance, enabling administrators to analyze trends, detect anomalies, and ensure smooth operation. Additionally, it provides the ability to **manage running processes dynamically**, allowing administrators to **start, pause, and terminate processes** in real time.

# Objectives

The primary goals of the Real-Time Process Monitoring Dashboard are:

* + **Real-Time System Monitoring:** Continuously tracks and displays **system resource usage** to provide a clear overview of system health.
  + **Graphical Representation:** Uses interactive **charts and visual indicators** to present

**CPU, memory, disk, and network usage** in an easily interpretable format.

* + **Process Management:** Allows administrators to **view and control active processes**, including the ability to terminate unresponsive or resource-heavy processes.
  + **User-Friendly Interface:** Designed with a **web-based UI** to ensure ease of access and seamless navigation for system administrators.
  + **Live Alerts for System Issues:** Implements a **real-time alert mechanism** to notify administrators about **critical issues**, such as excessive CPU or memory consumption, allowing for quick resolution.

# Technologies Used

The Real-Time Process Monitoring Dashboard is built using **a combination of backend and frontend technologies** to ensure efficient data processing and visualization.

# Backend Technologies:

* + **Python (Flask, psutil, Flask-SocketIO):**
    - **Flask:** A lightweight web framework that powers the dashboard’s backend.
    - **psutil:** A Python library that provides system monitoring capabilities, such as fetching CPU and memory usage statistics.
    - **Flask-SocketIO:** Enables real-time communication between the server and the client using WebSockets, ensuring seamless live updates.

# Frontend Technologies:

* + **HTML, CSS, JavaScript:** Used to design the dashboard's interface for displaying monitoring data effectively.
  + **Chart.js:** A JavaScript library that provides dynamic and interactive charts for representing resource usage visually.
  + **Socket.io:** Facilitates real-time updates between the client and server to keep monitoring data fresh without delays.

# Essential Libraries & Dependencies:

The following Python libraries are used for backend development:

* + Flask==2.2.2: Handles web application functionality and request routing.
  + psutil==5.9.0: Enables real-time system monitoring by retrieving CPU, memory, and disk usage statistics.
  + python-socketio==5.7.2: Supports WebSocket-based communication for instant data updates.
  + eventlet==0.33.0: Provides asynchronous networking support, ensuring smooth real- time interactions.

# System Design

The Real-Time Process Monitoring Dashboard is built using a modular approach, consisting of two main components:

# Backend Server (Python - Flask & SocketIO)

* + The backend is responsible for gathering system resource data using psutil, a Python library that provides real-time metrics related to CPU, memory, disk, and network usage.
  + This data is **streamed to the frontend** using WebSockets (Flask-SocketIO), ensuring administrators receive updates in real time without needing manual page refreshes.

# Frontend (HTML, CSS, JavaScript)

* + The frontend serves as a **visual interface**, presenting real-time system metrics using interactive graphs and dynamic tables.
  + Administrators can **monitor active processes** and **initiate management actions**, such as starting and terminating processes.

# Data Flow:

* The **backend continuously collects** real-time system metrics and active process details.
* This data is **transmitted via WebSockets** to the frontend.
* The **frontend dynamically updates** its graphs and tables to reflect real-time changes without requiring a page reload.
* Users can **control system processes**, such as terminating or starting new processes, using API requests handled by Flask.

# Implementation Backend (server.py)

* + Initializes a **Flask server** to handle requests and process system monitoring.
  + Uses psutil to collect real-time **CPU, memory, disk, and network usage** data.
  + Implements Flask-SocketIO to **stream live system metrics** to the frontend in real time.
  + Provides **API endpoints** for process management, allowing users to **start or terminate** system processes.

# Frontend (index.html, JavaScript)

* + **Visualizes system metrics** using Chart.js, a JavaScript library for creating interactive graphs.
  + Implements a **dynamic table** to display active running processes.
  + Provides **search and sorting functionalities**, enabling administrators to filter and locate specific processes efficiently.
  + Sends **requests to the backend** via API calls to manage system processes.

# Features

* + **Real-Time Performance Monitoring:**
    - Displays live graphs for **CPU, memory, disk, and network usage** to help administrators track system health dynamically.

# Process Management:

* + - Allows users to **start or terminate processes** directly from the dashboard.
    - Provides visibility into **process states and resource consumption**.

# Alerts & Notifications:

* + - Detects **high CPU or memory usage** and sends notifications, helping administrators react quickly to potential issues.

# Interactive UI:

* + - Supports **search, sorting, and filtering** for active processes, improving system navigation and usability.

# Web-Based Dashboard:

* + - Designed to be **accessible from any device** with a web browser, ensuring flexibility in system monitoring.

# Conclusion

The **Real-Time Process Monitoring Dashboard** is a robust and efficient tool designed to **enhance system monitoring and management**. By providing **real-time visual insights** into **CPU, memory, disk, and network usage**, it empowers administrators to **maintain system performance** and **quickly detect potential bottlenecks or issues**.

With **interactive graphs, dynamic process management, live alerts, and a web-based interface**, the dashboard simplifies **system resource tracking** and **optimizes workflow efficiency**. Its **backend, built with Flask and SocketIO**, ensures **seamless data transmission**, while the **frontend, powered by JavaScript and Chart.js**, delivers an intuitive user experience.

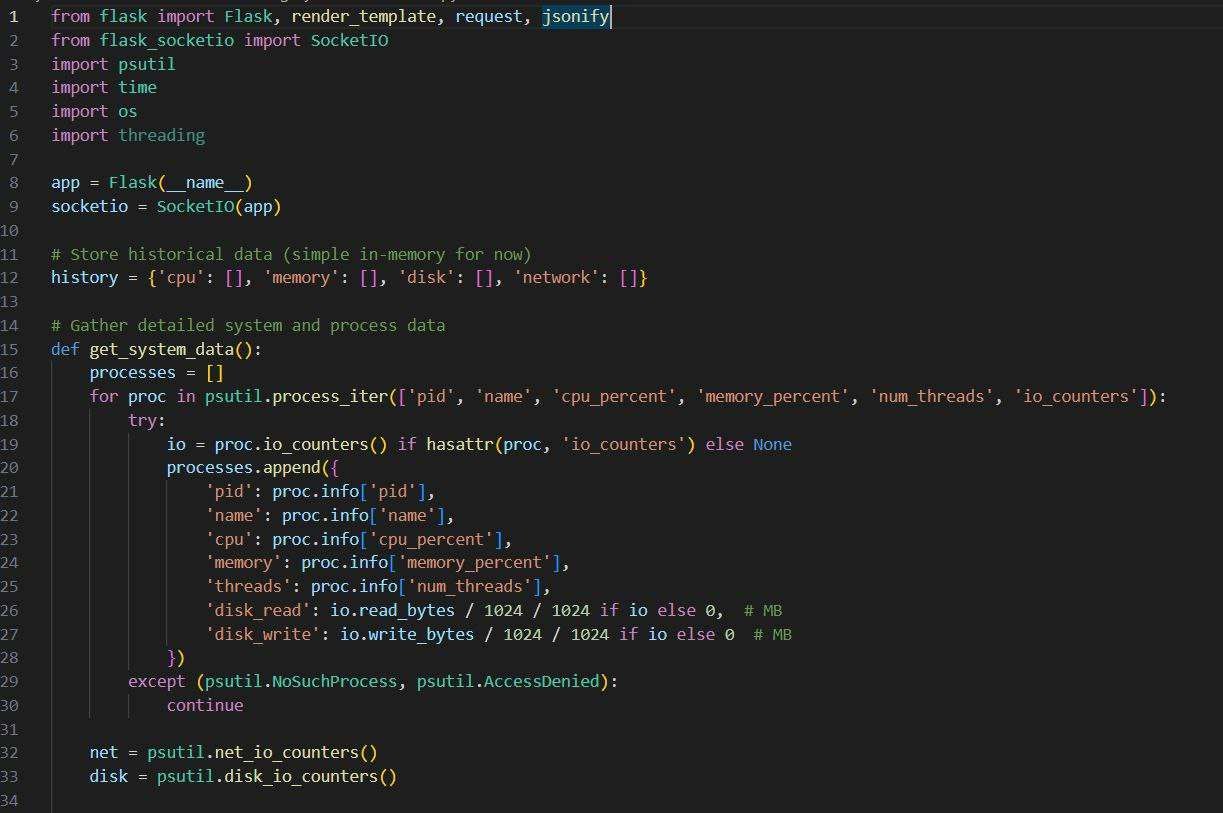
Looking ahead, potential **future enhancements**, such as **user authentication, historical data storage, process prioritization, and multi-system monitoring**, can further improve its capabilities and scalability.

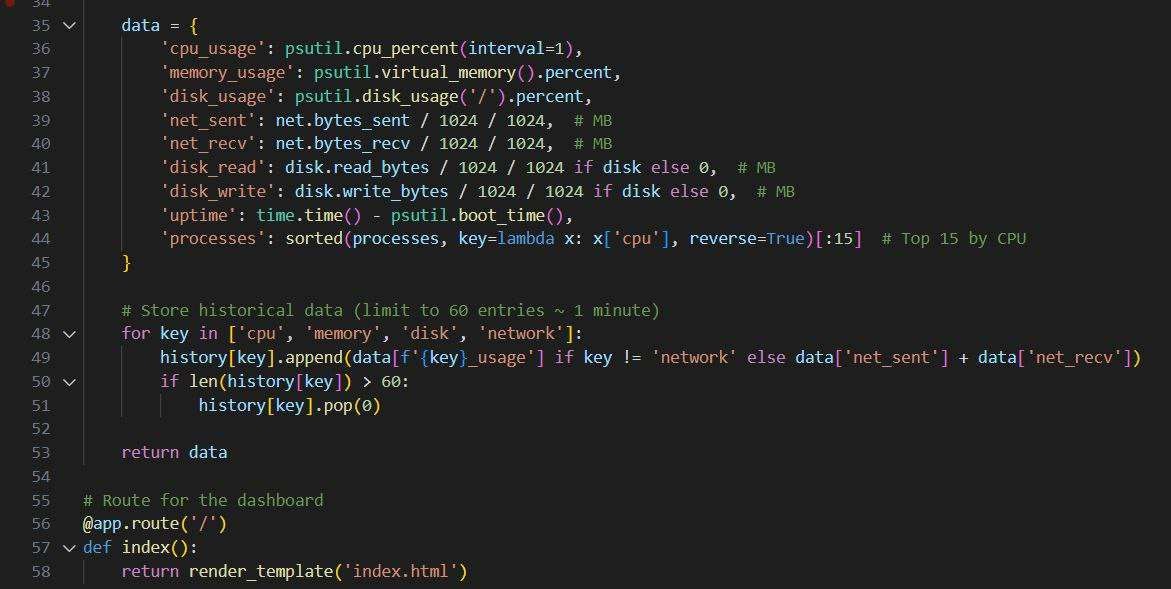
Ultimately, this dashboard serves as a **powerful asset** for system administrators, providing the necessary tools to ensure **real-time monitoring, proactive issue detection, and efficient system management**, all within an easily accessible web-based interface.

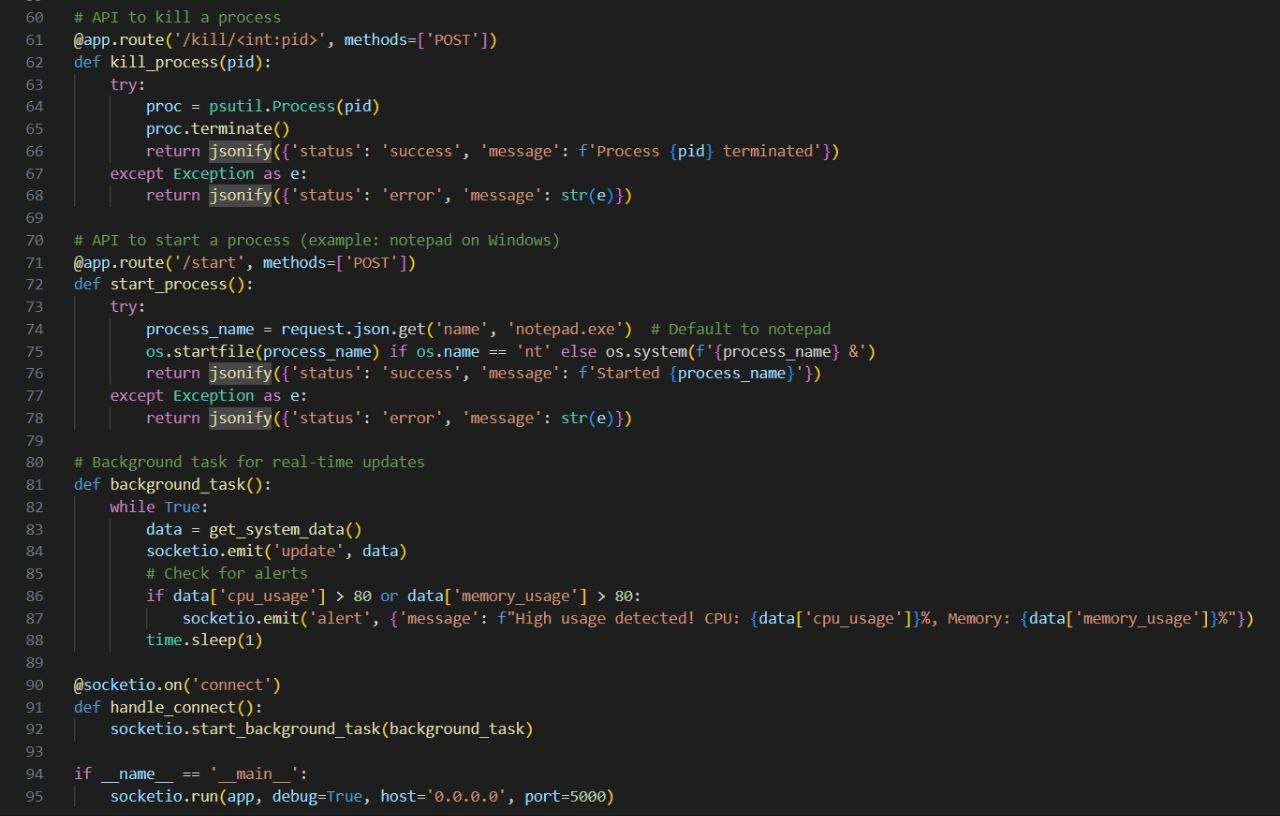
# References

* + Flask Documentation: https://flask.palletsprojects.com/
  + psutil Documentation: https://psutil.readthedocs.io/
  + Chart.js Documentation: https://[www.chartjs.org/](http://www.chartjs.org/)
  + Flask-SocketIO Documentation: https://flask-socketio.readthedocs.io/

*Here are Some Snapshot’s of Source Code*

**



**

