SALIM HABIB UNIVERSITY

(FORMERLY BARRETT HODGSON UNIVERSITY)

Project Report of Operating System

**************Title*************

Web Server Request Monitor

GROUP MEMBERS

DILSHAD ALI	**ID # F22CSC030**
FAISAL ALI	**ID # F22CSC023**
Hassan	**ID # S22CSC003**

DEPARTMENT # COMPUTER SCIENCE

Table of Contents

1. Project Overview

This project is a Web Server Monitoring System that helps track the performance and activity of a server in real-time. It records essential details like CPU usage, memory usage, total requests, average response time, and status codes for incoming requests. The goal is to assist server administrators in analyzing server performance and quickly identifying potential issues.

2. Introduction

2.1 Problem Statement

Web servers are critical components of modern digital infrastructure, hosting applications and services accessed by millions of users worldwide. Maintaining the performance, reliability, and efficiency of these servers is vital to ensure smooth operations. However, identifying issues such as high resource usage, frequent errors, or slow response times can be challenging without an effective monitoring system.

2.2 Proposed Solution

This project presents a Web Server Monitoring System designed to address these challenges by:

- ★Tracking system performance metrics like CPU and memory usage in real-time.
- ★Logging HTTP requests and responses for detailed analysis.
- ★Providing an intuitive, user-friendly dashboard for administrators to visualize server health and performance.

  The solution enables proactive problem identification and decision-making, ensuring servers operate at peak efficiency.

3. Methodology

3.1 Overview

The Web Server Monitoring System is implemented using Python with the Flask framework, supported by the `psutil` library for system monitoring. The project consists of two main components:

★ **app.py:** Handles HTTP requests, processes data, and serves the HTML dashboard.
★ **monitor.py:** Tracks system performance and logs request details.

3.2 Control Flow

★ **Server Initialization:**
  ○ The Flask server initializes and prepares variables to track performance metrics.
★ **Request Handling:**
  ○ **Pre-processing:** Log request start time and increment request counter.
  ○ **Processing:** Process the request, fetching or calculating the required data.
  ○ **Post-processing:** Log response details, update performance metrics, and store logs.
★ **Data Display:**
  ○ **JSON Data:** Available through `/metrics` and `/stats` endpoints for real-time monitoring.
  ○ **Dashboard:** Accessible at `/`, providing an overview of server health.

3.3 Key Features

★ **System Performance Monitoring:**
  ○ Metrics like CPU and memory usage are retrieved in real time using the `psutil` library.
★ **Request Tracking:**

    ○ Logs every HTTP request with details like method, URL, status code, and response time.

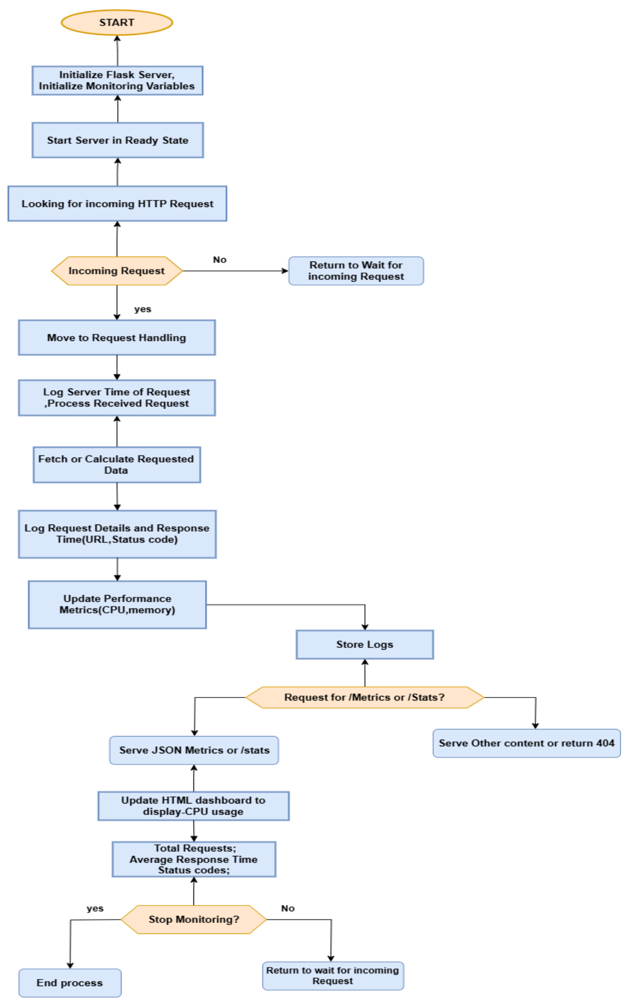
★ **Real-Time Statistics:**

    ○ Aggregates metrics like total requests, average response time, and status code counts.

★ **HTML Dashboard:**

    ○ Displays server health and links to detailed statistics in a clean, user-friendly interface.

3.4 Flow Chart


START
Initialize Flask Server, Initialize Monitoring Variables
Start Server in Ready State
Looking for incoming HTTP Request
Incoming Request
No
Return to Wait for incoming Request
yes
Move to Request Handling
Log Server Time of Request ,Process Received Request
Fetch or Calculate Requested Data
Log Request Details and Response Time(URL,Status code)
Update Performance Metrics(CPU,memory)
Store Logs
Request for /Metrics or /Stats?
Serve JSON Metrics or /stats
Serve Other content or return 404
Update HTML dashboard to display-CPU usage
Total Requests; Average Response Time Status codes;
Stop Monitoring?
yes
No
End process
Return to wait for incoming Request

4. Results

The system tracks CPU, memory usage, and HTTP request details, offering real-time metrics via JSON endpoints and an interactive dashboard for server performance analysis.

4.1 Logs

monitor_logs.txt

```
2024-11-22 10:00:00 - CPU Usage: 25.5%,

Memory Usage: 50.2%

2024-11-22 10:01:00 - Request: Method=GET,
URL=/metrics, Status=200, Response Time=120ms
```

4.2 JSON Responses

/metrics

```
{

    "cpu_usage": 23.5,

    "memory_usage": 55.8

}
```

/stats

```
{

    "total_requests": 5,

    "average_response_time": 120.5,


 "status_codes": {

        "200": 4,

        "404": 1

    }
}
```

4.3 Dashboard

- ★Displays an overview of system health and links to endpoints for detailed statistics.

5. Future Scope

- ★**Expanded Metrics**
  - Include additional system metrics such as disk usage, network activity, and I/O statistics.
- ★**Multi-Server Support**
  - Extend functionality to monitor multiple servers simultaneously.
- ★**Alerts and Notifications**
  - Integrate alerts for high CPU usage, memory consumption, or frequent errors.
- ★**Advanced Analytics**
  - Use machine learning models to predict performance bottlenecks and potential failures.

6. Conclusion

The Web Server Monitoring System demonstrates how to effectively monitor server performance in real-time. By leveraging Python, Flask, and `psutil`, the project provides a comprehensive solution for system performance tracking and request logging. The user-friendly dashboard ensures ease of use for administrators, enabling proactive problem-solving and efficient server management. The project’s scalability ensures it can be adapted to meet the evolving needs of modern server environments.