Logo, company name

Description automatically generated

International University of Sarajevo

CS307 Operating System

[**System Health Dashboard**](https://github.com/ardenfndi/System-Health-Dashboard)

Project Report

***Group Members***

Ethem Kesim 210302205

Emre Çubuk 220302303

Mustafa Atif Okutan 210302337

Ozan Umut Güney 200302109

# **Introduction**

The *System Health Dashboard* project is designed to monitor real-time system performance metrics, such as CPU usage, memory consumption, disk utilization, and network activity.  
The goal of this project is to create a web-based platform that provides users with a clear, dynamic, and interactive visualization of system health data.

This project demonstrates how backend data collection and frontend visualization can be integrated into a single, real-time system monitoring solution using modern web technologies.

# **Project Objectives**

•To design a lightweight monitoring agent that collects key system metrics.

•To build a backend API that communicates these metrics to a frontend dashboard.

•To display real-time performance metrics using interactive visual components.

•To provide a user-friendly interface for monitoring and diagnostics.

• To improve system transparency and allow users to identify potential issues early.

# **Technologies Used**

| **Component** | **Technology** | **Description** |
| --- | --- | --- |
| **Frontend** | HTML, CSS, JavaScript, React.js | Used to build an interactive and responsive dashboard interface. |
| **Backend** | Python (Flask Framework) | Handles API creation and real-time data streaming to the frontend. |
| **System Monitoring** | psutil Library | Gathers system metrics like CPU, memory, disk, and network usage. |
| **Real-Time Communication** | Socket.IO | Enables real-time data updates between the server and the client. |
| **Data Visualization** | Recharts (React Library) | Displays metrics in the form of graphs and charts. |
| **Development Tools** | VS Code, Node.js, Python Environment | Used for code development, dependency management, and testing. |

# **System Architecture**

The project follows a **client-server model** with two main components:

1. **Monitoring Agent (Backend):**

•Written in Python using Flask.

•Uses psutil to collect system performance data.

• Emits metrics via **Socket.IO** to the frontend.

1. **Dashboard (Frontend):**

•Built using React.js.

•Connects to the backend server using **socket.io-client**.

•Displays real-time data in graphs and charts for CPU, RAM, disk, and network usage.

# **Implementation Details**

## **Backend Implementation (Flask + psutil)**

The backend application continuously collects metrics such as:

* CPU usage percentage
* Memory utilization
* Disk read/write speed
* Network I/O (sent/received data)

These metrics are then broadcasted via a **Socket.IO** server to all connected clients.

## **5.2 Frontend Implementation (React + Socket.IO + Recharts)**

• The frontend establishes a socket connection to the Flask backend using socket.io-client.

• Data received through sockets is stored in state variables and visualized through charts.

• The dashboard interface typically includes:

- Line charts for CPU and memory usage over time

- Pie charts for disk usage

- Network traffic indicators

# **6. Key Features**

* **Real-time Monitoring:** Live data updates using WebSockets.
* **Cross-Platform:** Works on any OS that supports Python and Node.js.
* **Interactive Visualization:** Graphs automatically update to reflect system changes.
* **Modular Design:** Each metric is handled by a separate component for maintainability.
* **Scalable Architecture:** Can be extended to monitor multiple systems remotely.

# **7. Challenges Faced**

* Ensuring smooth and efficient data transfer between backend and frontend.
* Managing real-time updates without lag or data inconsistency.
* Handling cross-origin communication between Flask and React servers.
* Synchronizing multiple data streams simultaneously.
* Designing a clean, responsive, and informative dashboard layout.

# **8. Results and Performance**

The *System Health Dashboard* successfully:

* Displays accurate, real-time system metrics.
* Maintains efficient data flow with low latency.
* Provides a clear visual understanding of system performance.
* Uses minimal system resources for monitoring.

Performance testing confirmed that:

* CPU and memory readings are accurate within ±2%.
* The dashboard updates every second without noticeable delay.

# **9. Conclusion**

This project effectively integrates backend data processing and frontend visualization into a cohesive monitoring solution.  
It demonstrates the use of **Flask**, **Socket.IO**, and **React.js** to create real-time web applications.  
In future improvements, the project could be expanded to include:

* Historical data storage and trend analysis.
* Alert notifications when metrics exceed thresholds.
* Multi-system monitoring with authentication and user management.