**Dynamic Resource Monitor: Project Report**

Prepared by: Nidhidh Vyas

GitHub Repository - [github link](https://github.com/NidhishVyas/OS-Project)

**About the Project**

The Dynamic Resource Monitor project focuses on creating a comprehensive, real-time system monitoring application using Python for the backend and React.js for the frontend. This project integrates key system metrics like disk usage, network activity, system boot time, and top processes into a unified dashboard. The results provide a user-friendly, visually appealing interface that simplifies system monitoring for both developers and casual users.

**How the group has managed to complete the project**

Research Methods

The project began with extensive research on system monitoring libraries in Python, particularly psutil, and visualization tools like Chart.js for React. Research involved reading official documentation, exploring online forums, and reviewing similar projects for inspiration.

Data Collection Procedures

Data was collected using APIs hosted on a Flask backend that interfaced with the system’s real-time metrics. React.js was employed to fetch and render this data dynamically on the frontend.

Analysis Techniques

Metrics such as CPU usage, memory utilization, and network speeds were processed using Python. The data was analyzed for trends and abnormalities, with real-time graphs and charts providing a clear view of system performance.

**Key Findings**

The dashboard successfully displays critical system metrics, including:

•Disk Usage: Real-time disk utilization percentages.

•Network Speeds: Upload and download speeds in Mbps.

•System Boot Time: Detailed boot and uptime statistics.

•Top Processes: A ranked list of resource-intensive processes.

**Challenges Faced and Solutions**

1.Integration of Backend and Frontend:

Initially, it was challenging to synchronize the Flask backend with the React frontend. The solution was found by debugging API routes and leveraging tools like Postman to ensure the backend responses matched frontend expectations.

2.Real-Time Updates:

Maintaining real-time updates without overwhelming system resources was difficult. This issue was resolved by optimizing the data fetching interval to every 2 seconds and implementing efficient state management using React’s useState and useEffect hooks.

3.Data Visualization:

Presenting data dynamically using Chart.js required careful mapping of data points. Researching tutorials and community forums provided insights into configuring the library effectively.

4.Styling and Responsiveness:

Ensuring a clean and responsive design took several iterations. Styled Components and CSS Flexbox were used extensively to achieve the desired results.

**My Role in the Project**

I was responsible for integrating the backend with the frontend, ensuring seamless communication between the APIs and the React application. I also took the lead in designing the dashboard layout and configuring Chart.js for real-time visualizations.

I was responsible for collating all the work and ensuring the project was submitted on time. My teammate contributed equally in refining the final deliverables.

**Teammate Contributions**

•Yugandhar Patil: Focused on backend development, particularly the implementation of the psutil library and the Flask routes. He also ensured the accuracy of the data being fetched.

**References**

1.Python documentation: https://docs.python.org

2.React.js documentation: https://reactjs.org/docs

3.Chart.js tutorials: https://www.chartjs.org