Graphic Era deemed to be University DEHRADUN

PROJECT AND TEAM INFORMATION

Project Title

System Performance Analyzer – A Lightweight Monitoring Tool

Student/Team Information

Team Name: Team #	Architechs CD-VI-T155
Team member 1 (Team Lead) Jasuja, Harshit: 2021229 : harshitjasuja70@gamil.com	
Team member 2 Dixit, Yashika: 22022577 : yashikadixit1611@gmail.com	
Team member 3 Srivastava, Shivendra: 220211349 :	

PROJECT PROGRESS DESCRIPTION (35 pts)

Project Abstract (2 pts)

The System Performance Analyzer is a lightweight Python-based desktop application designed to monitor the real-time performance of systems. It displays vital system metrics such as CPU usage, RAM utilization, disk space, and other specifications using a clean graphical user interface developed with `tkinter`. The project addresses the lack of open-source performance tools optimized for macOS's native resolution and usability. The tool is tailored and includes user-centric features such as splash screens, alert popups, real-time charts via `matplotlib`, and logging/export functionality using JSON. It ensures responsiveness and performance using threading and provides a polished, accessible solution for students, developers, and everyday users.

Updated Project Approach and Architecture (2 pts)

Our system is designed in a modular fashion with the following components:

- User Interface Layer: Built using `tkinter`, this layer consists of multiple frames and widgets to display system stats, real-time graphs, and alerts.
- System Metrics Engine: Powered by the `psutil` library, this component gathers live data about CPU, memory, disk, and system information.
- Visualization Engine: Implements `matplotlib` for plotting data. `FigureCanvasTkAgg` is used to embed live graphs within the GUI.
- Thread Management: Background threads are used for periodic data fetching and updating to maintain GUI responsiveness.
- Utility Modules: Features such as splash screen initialization, export logs to JSON, and alert mechanisms for high resource usage.

All modules communicate internally through function calls and callbacks. No external server or network-based communication is needed.

Tasks Completed (7 pts)

Task Completed	Team Member
GUI design and layout	Yashika Dixit
Splash screen and branding integration	Yashika Dixit
psutil system data integration	Harshit Jasuja
Real-time graph plotting with matplotlib	Shivendra Srivastava
Threading and live data updates	Harshit Jasuja
Alert system for high CPU usage	Shivendra Srivastava

Challenges/Roadblocks (7 pts

One of the main challenges was **maintaining GUI responsiveness** while fetching and updating system data in real time. This was addressed using Python's `threading` module, ensuring that the mainloop of `tkinter` was not blocked.
Another issue was adapting the UI to **screen**, which required tweaking layout, resolution scaling, and widget spacing for optimal visibility.

Integrating `matplotlib` charts into `tkinter` was initially non-intuitive due to canvas compatibility issues. This was overcome by using `FigureCanvasTkAgg`.

Further, managing background threads safely to avoid data race conditions required synchronization strategies and cautious design. Despite these challenges, the team worked collaboratively to modularize the components and iteratively test each feature for smooth integration.

Tasks Pending (7 pts)

Task Pending	Team Member (to complete the task)
Add process-level tracking	Harshit Jasuja
Theme customization (light/dark mode)	Yashika Dixit
Packaging tool into .app (macOS)	Shivendra Srivastava
Additional alert thresholds (RAM, Disk)	Harshit Jasuja
Final UI polish and minor bug fixes	Entire Team

Project Outcome/Deliverables (2 pts)

The primary outcome is a **desktop-based application** that provides:

- Real-time graphs of CPU, RAM, and disk utilization.
- A user-friendly GUI with splash screens and alert systems.
- Exportable JSON logs of performance data.
- A scalable, modular framework adaptable for future additions like cloud syncing and process-level monitoring.

Progress Overview (2 pts)

Approximately **75-80% of the project** is completed. All core features—UI, performance data monitoring, graph plotting, alerts, and export—are functional. UI refinement and packaging are underway. We are ahead of schedule in terms of feature development and plan to wrap up with testing and minor enhancements in the coming week.

Codebase Information (2 pts)

Repository Link: [Insert GitHub/Repo Link]
Branch: `main`
Notable Commits:

- Splash screen integration: `commit#e12f3a1`
- Graph plotting: `commit#c5d4f77`
- Export module: `commit#a8b9902`

Testing and Validation Status (2 pts)

Test Type	Status (Pass/Fail)	Notes
UI Responsiveness	Pass	Validated using simulated load
Metric Accuracy	Pass	Values verified with Activity Monitor
Alert Popup Functionality	Pass	Alerts triggered at correct thresholds
Export JSON Logs	Pass	File content verified manually

Deliverables Progress (2 pts)

System Monitoring Core (CPU, Memory, Disk) – Completed
Splash Screen & UI Framework – Completed
Real-Time Graph Plotting (matplotlib) – Completed
Data Logging & Export (JSON) – Completed
Alert System for Resource Thresholds – Completed
Settings Panel (e.g., refresh rate) – In Progress
Final Deployment Package (Executable/App) - In Progress