Bahria University, Islamabad

Department of Software Engineering

Operating Systems Lab

(Fall-2025)

Teacher: Engr. Saad Mazhar Khan

Student: Hasan Zahid

Enrollment: 01-131232-028

Lab Journal: Project Report

Date: December 17, 2025

GitHub Repository Link:

https://github.com/hasanzafzal/OS-Semester-Project

Comments:

Signature

LAB

SYSTEM HEALTH MONITORING AND VISUALIZATION SYSTEM

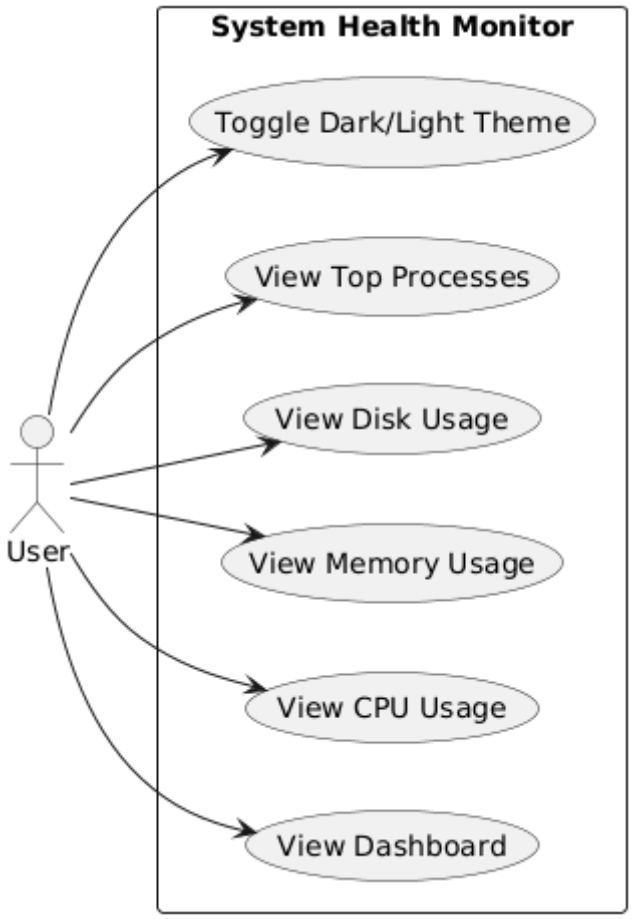
1. Introduction

- Problem Statement:
  Modern operating systems run multiple processes simultaneously, consuming CPU, memory, disk, and other system resources. Users often lack a simple and unified tool to monitor system health in real time.
- Objectives of the Project:
  - Monitor CPU, memory, disk usage and uptime
  - Implement backend monitoring using Bash scripting
  - Develop a GUI frontend using PyQt6
  - Display top CPU-consuming processes and real-time graphs
  - Demonstrate core Operating System concepts
- Scope and Limitations:
  Scope includes Fedora Linux systems with real-time monitoring and visualization. Limitations include lack of remote monitoring, process control, and persistent database storage.
- Significance of the Project:
  The project bridges CLI and GUI monitoring, helps understand OS resource management, and provides practical Linux administration skills.

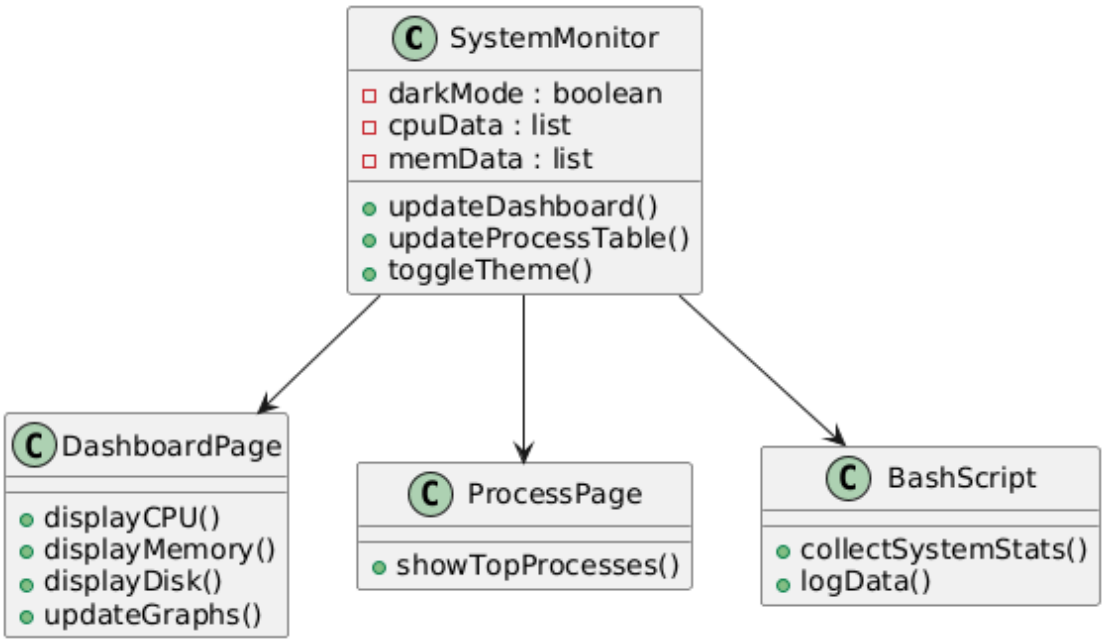
2. System Analysis

- Requirement Analysis:
  - Functional requirements include real-time monitoring, process listing, and theme toggling.
  - Non-functional requirements include stability in Fedora 43 and responsiveness.
- Feasibility Study:
  - Technical feasibility is ensured by using open-source tools.
  - Operational feasibility is high due to user-friendly interface.
  - Financial feasibility is ensured with zero cost.

- Use Case Diagram:

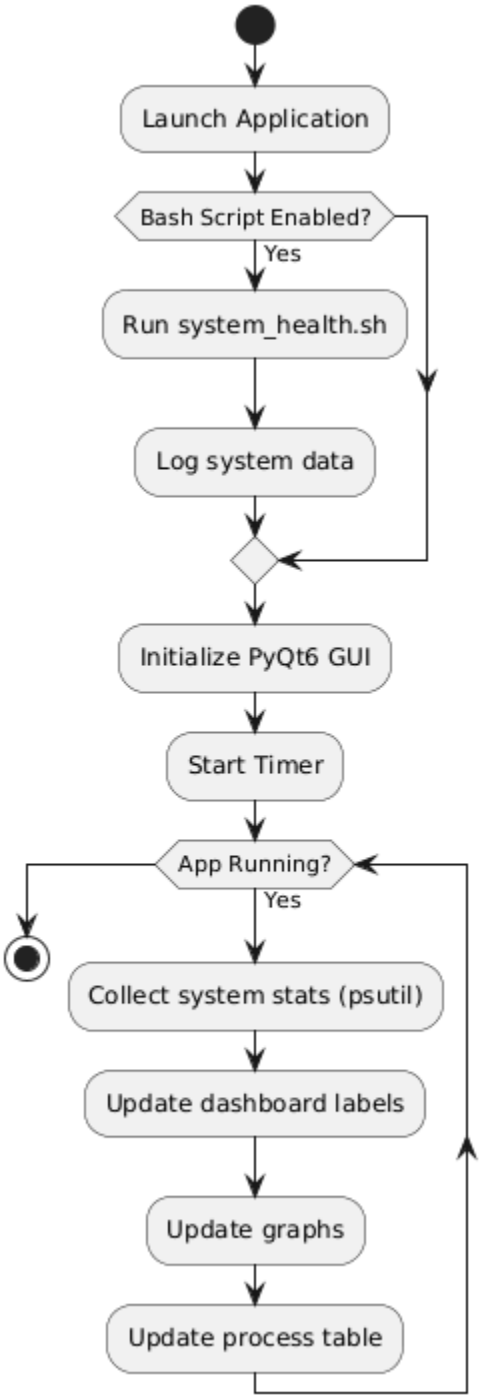


- Class Diagram:



3. System Design

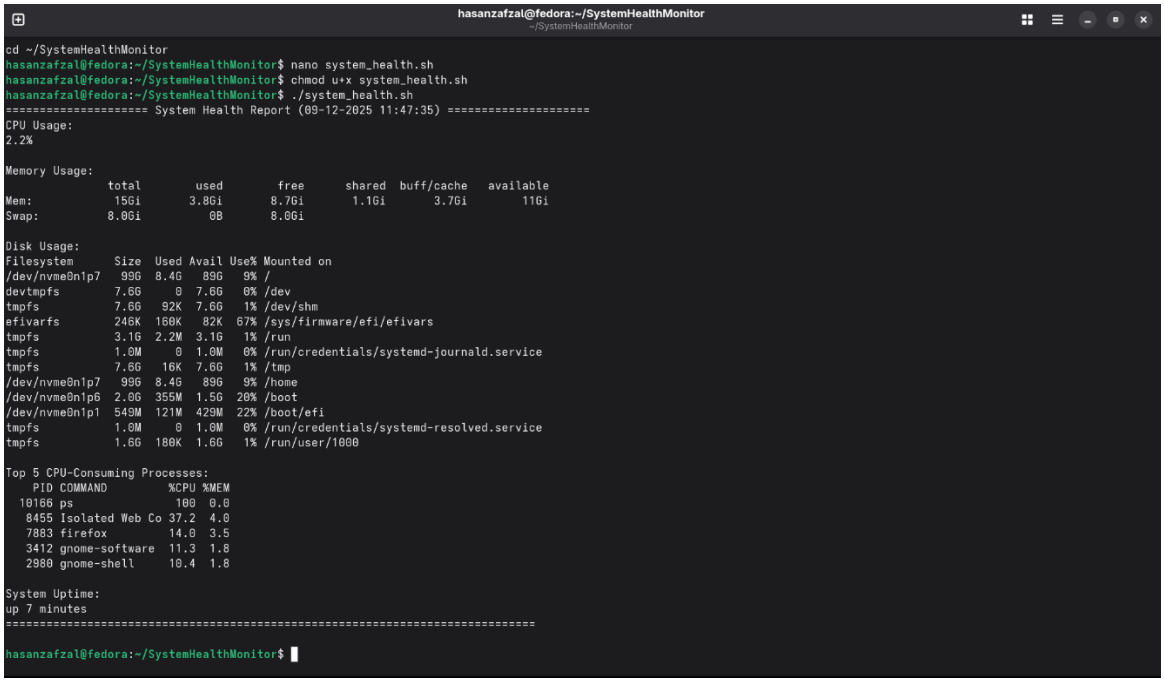
- Architectural Design:
  - The system follows a hybrid Bash backend and PyQt6 frontend architecture.
  - Flow-Chart:

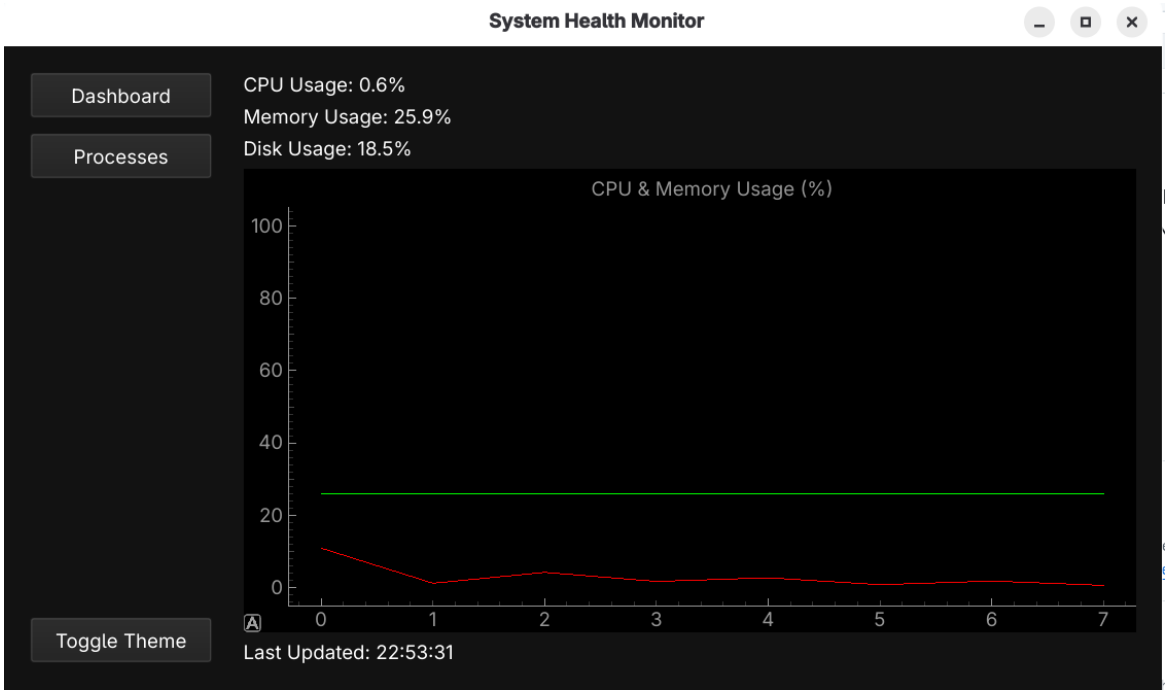


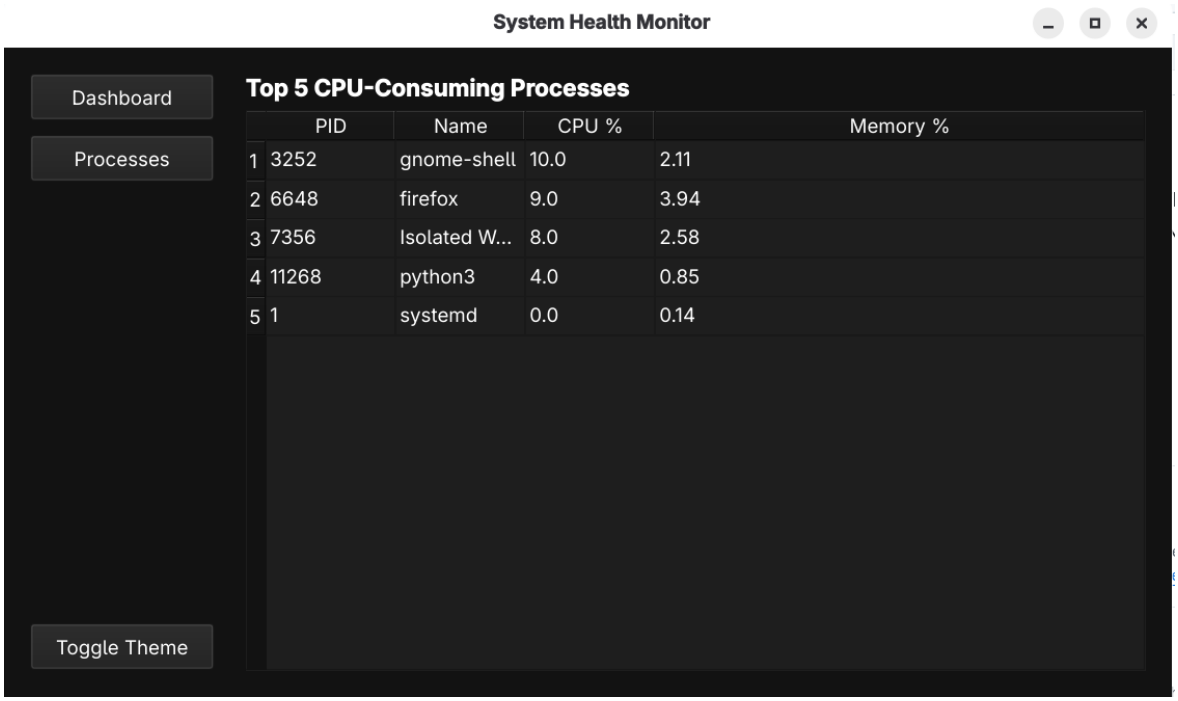
- Database Design:
  No database is used; logging is done via text files.
- User Interface Design:
  The UI includes a dashboard, graphs, process tables, and dark/light mode.

4. Implementation

- Programming Languages and Tools Used:
  - Bash
  - Python (PyQt6)
  - psutil
  - pyqtgraph
- Modules Description:
  Includes monitoring module, GUI dashboard, process viewer, and graph module.
- Code Snippets:
  Complete code is available on GitHub. (Link attached above)
- Screenshots:







5. Results and Discussion

- Outcomes:
  A stable, real-time system monitoring application was developed.
- Challenges Faced:
  - Creating GUI with bash program
  - GUI incompatibility issues on Fedora 43 were resolved using PyQt6.
- Comparison with Existing Solutions:
  The project provides GUI and graphs unlike traditional CLI tools.

6. Conclusion and Future Work

- Summary of Achievements:
  Successfully implemented a hybrid system monitoring solution.
- Future Enhancements:
  - Alerts
  - Historical data storage
  - Network monitoring
  - RPM packaging

7. References

- Fedora Project:
  Fedora Linux Documentation.

Available at: https://docs.fedoraproject.org
(Accessed for Fedora 43 system behavior, package management, and Wayland environment)

- GNU Bash Manual:
  Bash Reference Manual.
  Available at: https://www.gnu.org/software/bash/manual/
  (Referenced for shell syntax, execution permissions, and scripting practices)

- Python Software Foundation:
  Python 3 Documentation.
  Available at: https://docs.python.org/3/
  (Used for Python language constructs and subprocess integration)

- psutil Development Team:
  psutil – Cross-platform system and process utilities.
  Available at: https://psutil.readthedocs.io
  (Used for retrieving CPU, memory, disk usage, uptime, and process statistics)

- PyQt Development Team (Riverbank Computing):
  PyQt6 Documentation.
  Available at: https://www.riverbankcomputing.com/static/Docs/PyQt6/
  (Used for GUI design, widgets, layouts, timers, and styling)

- pyqtgraph Developers:
  pyqtgraph Documentation.
  Available at: https://pyqtgraph.readthedocs.io
  (Used for real-time CPU and memory graph visualization)