Final Report
Operating System Project

Department Of Software Engineering
SMIU

Multi-OS Process Monitoring Scheduling Analysis System

Project Leader Muhammad
BSE-23F-098

Student-1 Haram Zainab
BSE-23F-137

Student-2 Syeda Sanawar Kamran
BSE-23F-102

Assigned By: Sir Asif Irshad

1. Motivation

The motivation behind developing the Multi OS Process Monitoring Scheduling Analysis System is to gain a deeper practical understanding of how modern operating systems manage and schedule processes internally. While operating system concepts such as process management, scheduling, and resource utilization are often studied theoretically, their real-world behavior across different platforms remains complex and less visible to users. This project was undertaken to bridge that gap by providing a unified system capable of monitoring processes across multiple operating systems. Additionally, from a security perspective, monitoring running processes plays a crucial role in identifying suspicious or harmful activities within a system. By designing a prototype that observes and analyzes process behavior across Windows, Linux, and Android, this project demonstrates how such monitoring tools can contribute to system transparency, learning, and security awareness, while also laying the foundation for a scalable and more comprehensive solution in the future.

2. Overview

2.1 Significance/Importance of project

The Multi OS Process Monitoring Scheduling Analysis System holds significant academic and practical value as it addresses a core component of operating systems: process management. Organizations, developers, and system administrators require effective tools to observe what is running within their systems to ensure stability, performance, and security. This project provides an intuitive way to monitor processes and understand their behavior without requiring deep technical expertise. By enabling process visibility across multiple operating systems, the system enhances learning, assists in detecting potentially harmful processes, and supports better system control. Its practical implementation strengthens core operating system concepts and demonstrates their application in real-world environments.

2.2 Description of project

The Multi OS Process Monitoring Scheduling Analysis System is a cross-platform application designed to monitor and analyze running processes on Windows, Linux, and Android operating systems. Upon launching the system, users are presented with an interface that allows them to select the desired operating system. Once an operating system is selected, the system retrieves and displays a list of all active processes running on that platform. When a user selects a specific process, detailed information is shown, including process identifiers, resource utilization, execution statistics, and scheduling-related data. The system updates process information in real time, allowing users to observe dynamic changes in resource usage and process behavior. The project adopts a generalized approach, making it applicable to multiple environments rather than a single system-specific use case.

2.3 Related Work and Literature Review

The development of this project was guided by a combination of academic knowledge and practical exploration. Various online learning resources were consulted to understand process monitoring techniques and system-level programming. Platforms such as YouTube were used for conceptual demonstrations and implementation guidance related to operating systems, Python programming, and process management. ChatGPT was utilized as an interactive learning and problem-solving assistant to refine implementation approaches and clarify technical concepts. General research through Google supported understanding of operating system utilities and behavior. The project itself was primarily self-developed, following the guidance and direction provided by the course instructor, without direct reuse of third-party implementations.

3. Salient Features

- **Multi-Operating System Support**

  The system is designed to work across multiple operating systems including **Windows, Linux, and Android** through a single unified interface. Unlike existing process monitoring tools that are limited to a single platform, this project provides cross-platform compatibility, making it more versatile and widely applicable for users, students, and evaluators.

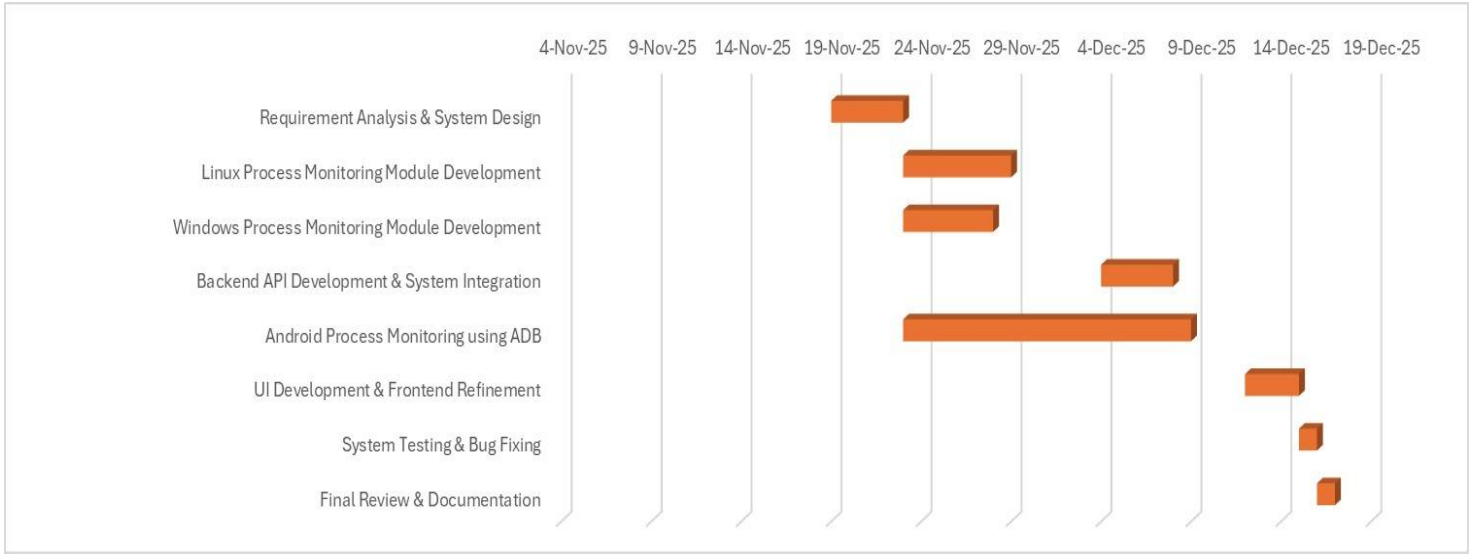
- **Process Purpose Identification**

  A unique feature of this system is that it displays not only technical process details but also the purpose of each running process**.**
  In **Windows**, some process purposes are obtained directly from the operating system, while additional purposes are provided through hard-coded mappings.
  In **Linux and Android**, where the operating systems do not provide explicit process purposes, all process purposes are manually mapped within the system.
  This feature helps users understand why a process is running and what function it performs**,** which is not commonly available in standard monitoring tools.

3. Project Planning

The project was developed over a structured timeline spanning from **19 November to 16 December**. During this period, responsibilities were distributed among team members to ensure efficient development and parallel progress. The Linux process monitoring module was designed and implemented by **Syeda Sanawar Kamran**, focusing on process retrieval, analysis, and real-time updates. The Windows process monitoring module, backend API development, and core system integration were handled by **Muhammad**, ensuring accurate data collection and seamless communication between components. The Android process monitoring functionality and user interface development were carried out by **Haram Zainab**, integrating Android process data through ADB and refining the frontend experience. This structured division of tasks allowed the project to progress systematically from research and design to implementation, testing, and final documentation.

Gantt Chart:



5. Required Hardware and Software

• Hardware Requirements

The following hardware resources were required for the successful completion of the project:

Hardware Component	Specification
Processor	Intel Core i5 or equivalent
RAM	16 GB
Storage	Minimum 256 GB SSD
Operating Systems for Testing	Windows 10 Pro, Ubuntu Linux, Android Smartphone
Android Device	Redmi 14C (Android 15)
Connectivity	USB cable for ADB connection

These hardware resources were arranged using personal development systems and physical Android devices available to the team members.

• Software Requirements

The following software tools and platforms were used during development and testing:

Software	Purpose
Windows 10 Pro (22H2)	Windows process monitoring and testing
Ubuntu 24.04.3 LTS	Linux process monitoring
Android 15	Android process monitoring
Python 3.13.5	Backend development and process analysis
Visual Studio Code	Code development and debugging
Android Debug Bridge (ADB)	Android process data retrieval
Git	Version control and collaboration

All software tools were obtained through **official and open-source platforms**, ensuring reliability and easy availability.

• Programming Languages and Libraries

Programming Language: Python 3.13.5

Libraries Used:

- psutil for process monitoring on Windows and Linux
- subprocess for executing system-level commands
- platform for OS detection
- flask for backend API communication
- json for structured data handling
- These libraries were installed using the pip package manager.

• Configuration and Deployment of Other Systems

The project can be executed on any other compatible system by following these steps:

- Install **Python**.
- Install required Python libraries using pip install -r requirements.txt.

- Ensure the target system is **Windows, Linux, or Android**.
- For Android monitoring, enable **USB Debugging** on the device and install **ADB**.
- Run the backend server and platform-specific agent scripts.

This configuration ensures portability and allows the system to run smoothly on other machines with similar specifications.

6. Libraries / Modules / API used in Project (Third Party APIs Not Allowed)

- ## Description of Libraries Used in the Project
  - **psutil (v6.x)**

    Used to retrieve process and system information such as CPU usage, memory consumption, running processes, threads, and I/O statistics on Windows and Linux platforms.
    Reference: https://pypi.org/project/psutil/

  - **ctypes (Built-in, Python 3.13.5)**

    Used to access low-level Windows API functions by interacting with system DLLs such as kernel32 and version for enhanced process monitoring.
    Reference: https://docs.python.org/3/library/ctypes.html

  - **wmi (v1.5.x)**

    Used to interact with Windows Management Instrumentation (WMI) to obtain detailed system and process-related information on Windows.
    Reference: https://pypi.org/project/WMI/

  - **subprocess (Built-in, Python 3.13.5)**

    Used to execute external commands. In this project, it is mainly used to run ADB commands for Android process monitoring.
    Reference: https://docs.python.org/3/library/subprocess.html

  - **FastAPI (v0.115.x)**

    Used to develop a high-performance RESTful backend API for communication between the Python backend and the Flutter frontend.
    Reference: https://fastapi.tiangolo.com/

  - **requests (v2.32.x)**

    Used to handle HTTP requests and responses for backend communication.
    Reference: https://pypi.org/project/requests/

  - **Standard Python Modules**

    Modules such as os, platform, datetime, time, socket, and json are used for system interaction, OS detection, time handling, networking, and data formatting.
    Reference: https://docs.python.org/3/library/

7. References

[1] Course instructor guidance and lecture material on Operating Systems.

[2] Python Official Documentation.

[3] psutil Library Documentation.

[4] Android Debug Bridge (ADB) Documentation.

[5] Online educational resources and tutorials accessed via YouTube.

[6] ChatGPT, OpenAI, used as a learning and development assistance tool.