NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(Peshawar CAMPUS)
FAST School of Computing
Spring 2025

Project Proposal: **[Mini Task Manager (Linux Edition)]**

1. Abstract:

The **Mini Task Manager** project is inspired by operating system utilities like **Task Manager in Windows** or **htop in Linux**, designed to provide real-time monitoring and management of system processes. The need for this tool arises from a common user requirement: managing system performance, terminating resource-heavy processes, and understanding how the operating system handles process scheduling and memory allocation.
This project will simulate a **lightweight system task manager**, capable of displaying currently running processes, their CPU and memory usage, and allowing the user to terminate selected processes. The main goal is to understand and implement OS concepts like **process management**, **multithreading**, **semaphores**, **system calls**, and **memory handling**.
By the end, the project will be a **CLI or GUI-based live monitoring tool** with process control functionalities, making it both educational and functional.

2. Group Members:

[Muhammad Taha] - [23P-0559]

[Abdul Rafy] - [23P-0559]

3. Project Overview

3.1 Objectives

- To simulate the functionality of a modern Task Manager.
- To allow users to view and sort processes by CPU and memory usage.
- To implement process termination by PID.
- To gain practical experience with key OS concepts.

3.2 Scope

- Real-time monitoring of processes and their resource usage.
- User ability to sort, search, and kill processes.
- Implementation of OS-level synchronization using semaphores.

- Bonus: Auto-kill high-resource processes or alert the user.

4. Functional Requirements

- Display list of running processes with CPU and memory stats.
- Refresh process list every second (multithreaded).
- Enable manual termination of any process by PID.
- Allow sorting based on CPU or memory usage.
- Optional: Search functionality for a specific process.

5. Non-Functional Requirements

- Should have an easy-to-use UI (CLI or GUI).
- Should refresh without lag (real-time updates).
- Code should be well-commented and modular.
- Should work on Linux systems (Ubuntu preferred).

6. Technologies and Tools

- Programming Language: Python (for simplicity) or C (for low-level syscalls)
- Libraries: psutil (for process monitoring), threading, tkinter (optional GUI)
- OS Concepts:
- Process management
- Threads and synchronization (semaphores)
- Memory handling
- System calls (kill, os.fork, exec, /proc usage in Linux)

7. Timeline and Milestones

Day Milestone
Day 1 Finalize idea, split tasks, install tools (Python, psutil, etc.), create GitHub repo
Day 2 Implement real-time process listing using psutil, print to CLI
Day 3 Add memory & CPU usage + sorting options (by CPU or RAM)
Day 4 Add "Kill Process" feature using os.kill() and semaphores (for safe multi-threading)
Day 5 Add search feature by PID/name, optional: auto-kill high-usage processes
Day 6 Polish UI (CLI interface or simple GUI using tkinter), test on Ubuntu
Day 7 Final report writing, flowchart, screenshots, and viva prep 💯

8. Expected Outcomes

A fully functional Mini Task Manager with real-time capabilities.
Deep understanding of core OS concepts like threading, semaphores, and process control.
An educational yet practical tool that demonstrates how operating systems manage tasks internally.
A neat, clean, and demo-ready project with documentation and flowcharts.