



Project Title:

CPU Scheduling Algorithm Simulator



Project Objectives:

- Understand how CPU scheduling works in an operating system.
 - Implement various scheduling algorithms: **FCFS, SJF, SRT, RR, and MLFQ**.
 - Simulate the behavior of these algorithms on a set of user-defined processes.
 - Display scheduling results using **Gantt chart**, and provide key metrics: **Waiting Time, Turnaround Time, Response Time**.
 - Compare performance between different scheduling strategies.
-



Project Requirements:

◆ Functional Requirements:

1. Process Input:

- Each process must have:
 - Process ID
 - Arrival Time
 - Burst Time
 - (Optional: Priority for MLFQ)
- Input can be from:
 - Console/CLI
 - File (e.g., CSV or JSON)

- (Optional) Simple GUI Form

2. Algorithms to Implement:

- **First Come First Serve (FCFS)**
- **Shortest Job First (SJF) – Non-preemptive**
- **Shortest Remaining Time (SRT) – Preemptive**
- **Round Robin (RR)** – with configurable time quantum
- **Multilevel Feedback Queue (MLFQ)** – basic 3-level queue with:
 - Different time quantum per queue
 - Promotion/demotion logic
 - Aging to prevent starvation

3. Simulation Output:

- Display Gantt chart for process execution
- Table with:
 - Waiting Time
 - Turnaround Time
 - Response Time
- Average values of the above metrics
- Option to export results (Optional)

4. User Interface:

- Console-based UI is acceptable
- Bonus: GUI using Tkinter, web-based interface, etc.



Sample Scenario:

Processes:

P1: Arrival=0, Burst=5

P2: Arrival=1, Burst=3

P3: Arrival=2, Burst=8

P4: Arrival=3, Burst=6

Quantum (RR, MLFQ): 2

MLFQ: 3 Queues with Quantum = [2, 4, FCFS]



Suggested Technology Stack:

- **Language:** Python, Java, C++, or JavaScript
 - **UI (Optional):** Tkinter (Python), Web (React/Vanilla JS), JavaFX
 - **Visualization (Optional):** matplotlib (Python), ASCII Gantt chart, HTML/CSS
-



Deliverables:

1. **Source Code**
2. **README file** with:
 - Setup instructions
 - Description of algorithms implemented
 - How to run each scheduler
3. **Sample Input/Output**
4. **Screenshots or Gantt chart output**
5. (Optional) **Report** explaining logic, challenges, and comparison of results