The Superior University

📝 Operating Systems Lab – Project Documentation Template

📌 Project Title

CPU Scheduling Simulator in PYTHON

👥 Group Members

- Ahsan Ullah-222
- Muhammad Umair-200
- Waleed khan-218
- Ali Hussnain-067

📂 GitHub Repository

GitHub Repository Link:

https://github.com/muhammadahsaanullah/CPU-Scheduling-simulator.git

---

🔧 Scheduling Algorithm Implemented

✅ Tick the scheduling algorithm your group implemented:

- ✅FCFS (First Come First Serve)
- ☐ SJF (Shortest Job First – Non-Preemptive)
- ☐ SJF (Preemptive)
- ☐ Round Robin

📄 Project Description

Briefly explain:

- This project simulates how a CPU schedules tasks using the First-Come-First-Serve (FCFS) algorithm. It helps visualize and calculate how processes are handled over time, optimizing resource planning and performance analysis.

- **Inputs:** Arrival Time and Burst Time for each process.
- **Outputs:** Start Time, Completion Time, Turnaround Time, Waiting Time, Average TAT and WT.
- **Implementation:** Processes are sorted by arrival time and executed in that order, with each process's times calculated sequentially.

📸 Output Screenshots

```
Enter the number of processes: 2

Enter details for Process 1
Arrival Time: 0
Burst Time: 3

Enter details for Process 2
Arrival Time: 1
Burst Time: 4
```

```
Final Process Table:
PID       Arrival   Burst     Start     CT        TAT       WT
P1        0         3         0         3         3         0
P2        1         4         3         7         6         2
```

```
Average Turnaround Time: 4.50
Average Waiting Time: 1.00
```

🧠 Code Structure & Explanation

- **Code Organization:** The code is organized using functions (`fcfsScheduling`, `printResults`, `printGanttChart`, and `main`) for modularity and clarity.
- **Core Logic:** The `fcfsScheduling` function sorts processes by arrival time and calculates start, completion, turnaround, and waiting times sequentially.

🛠️ Challenges Faced

1. Incorrect Waiting Time Calculation: Initially, we miscalculated waiting time by not considering idle CPU time; we fixed it by ensuring `current_time` jumps to the process's arrival time if the CPU is idle.

3. Sorting Logic Bug: Some processes were executed out of order due to not sorting by arrival time; we added a custom comparator to sort the process list before scheduling.c.