# A black and red text Description automatically generated

**PROJECT AND TEAM INFORMATION**

## Project Title

|  |
| --- |
| System Lite Enhancer |

## Student / Team Information

|  |  |
| --- | --- |
| Team Name:  Team # (Mentor needs to assign) | QuantumTrio |
| **Team member 1 (Team Lead)** | Baliyan, Khushi– 230211194 khushibaliyan026@gmail.com |
| **Team member 2** | Bisht, Nishant– 23011869  Nishantbisht08@gmail.com |
| **Team member 3** | Jain, Siddharth– 230211065  Siddharth2264@gmail.com |

**PROPOSAL DESCRIPTION (15 pts)**

## Motivation (2 pts)

|  |
| --- |
| Efficient CPU scheduling is very important for the performance and fairness of an operating system, especially in multiprogrammed environments. Traditional algorithms like **FCFS, SJF, Priority, and Round Robin** each have their own advantages, but also clear drawbacks:   * **FCFS** can cause the convoy effect (slow processes delay faster ones). * **SJF** requires accurate burst time prediction and may be unfair to longer jobs. * **Priority Scheduling** can lead to starvation of lower-priority processes. * **Round Robin** depends too much on the choice of time quantum, which can affect performance.   Recent research has tried to improve these algorithms by using dynamic time quantums or hybrid approaches. However, a simple yet effective hybrid method called **Modified Priority Preemptive (MPP) Scheduling**—which combines **Priority Preemptive Scheduling** with **Round Robin**, where the quantum is set to the shortest burst time among ready processes—has not been deeply explored.  This motivates our project: **to design, implement, and test a “System Lite Enhancer” tool that integrates the MPP algorithm, compares it with classical algorithms, and provides an easy-to-use interface for learning and analysis.** |

## State of the Art / Current solution (2 pts)

|  |
| --- |
| Classical scheduling algorithms like **FCFS, SJF, Priority, and Round Robin (RR)** are still widely used as the baseline in both teaching and research. They are simple to understand and implement but face limitations in efficiency and fairness.  **Recent Research:**   * A 2024 review in *Heliyon* (Singh & Goyal) shows that most new proposals are **RR-based hybrids**, often combined with **SJF or Priority**, with a strong focus on **dynamic quantum selection** to improve performance. * A 2020 study in *Procedia Computer Science* (Kumar & Kumar) proposed **SJF-RR** and **Priority-RR hybrids**, which showed better turnaround and waiting times. However, these still faced challenges like **starvation** and **increased overhead**.   **Research Gap:**   * In many studies, including the *Procedia 2020* paper, it is assumed that **all processes arrive at time 0**, which is not realistic for real-world systems. * The experiments were also run on **very small datasets**, making it hard to judge scalability and practical performance. * Very few works explore a **Modified Priority Preemptive (MPP) Scheduling algorithm**, where **Priority Preemptive Scheduling** is combined with **Round Robin**, and the **time quantum is set to the shortest burst time in the ready queue**.   This simple yet effective approach aims to:   * Reduce waiting time, * Improve fairness, and * Prevent starvation of low-priority processes.   These research gaps directly inspired our project, the **System Lite Enhancer**, which not only implements and tests MPP scheduling but also compares it with classical algorithms through tables, Gantt charts, and graphs, using flexible datasets with varying arrival times. |

## Project Goals and Milestones (3 pts)

|  |
| --- |
| **Project Goals**   1. **Implement baseline scheduling algorithms**: FCFS, SJF (non-preemptive and SRTF), Preemptive Priority, and Round Robin. 2. **Design and implement the proposed hybrid algorithm — Modified Priority Preemptive (MPP) Scheduling**: combines Preemptive Priority with Round Robin, using the **shortest burst time in the ready queue as the quantum**. 3. **Develop a UI-based simulator** to input process details (arrival time, burst time, priority), visualize Gantt charts, and generate comparison tables. 4. **Solve the research gap** by testing MPP and classical algorithms on **realistic datasets with varied arrival times**, not assuming AT = 0, and using larger, diverse workloads. 5. **Evaluate and compare performance** of all algorithms based on turnaround time, waiting time, response time, throughput, context switches, and fairness.   **Project Milestones**  **1. Implementation of Classical Algorithms with UI**   * Build and test FCFS, SJF (non-preemptive and SRTF), Preemptive Priority, and Round Robin scheduling in the simulator. * Provide a **graphical interface** to enter process details and automatically generate **Gantt charts** and result tables.   **2. Design and Integration of Proposed Hybrid Algorithm (MPP)**   * Implement the **Modified Priority Preemptive Scheduling (MPP)** algorithm: Preemptive Priority + Round Robin, with quantum = shortest burst time. * Compare its performance against classical algorithms using key metrics (waiting time, turnaround time, response time, throughput, fairness, and context switches).   **3. Addressing Research Gap**   * Test MPP and other algorithms with **flexible arrival times and larger datasets**, to ensure realistic, practical evaluation.   **4. Final Evaluation and Report**   * Analyze results, highlight improvements of MPP over existing algorithms, and prepare the final project report with graphs, tables, and conclusions. |

## Project Approach (3 pts)

|  |
| --- |
| **Project Approach**  **Design & Implementation Plan:**   1. **Data Model:** Represent each process as an object with attributes like:    * **PID** (Process ID)    * **AT** (Arrival Time)    * **BT** (Burst Time)    * **RT** (Remaining Time)    * **Priority**    * Other relevant metrics (waiting time, turnaround time, etc.) 2. **Scheduler Core:** Create a **scheduler interface** that can handle multiple algorithms:    * FCFS    * SJF (non-preemptive and SRTF)    * Preemptive Priority    * Round Robin    * **Modified Priority Preemptive (MPP) / PPRR** 3. **PPRR / MPP Algorithm Details:**    * Always choose the **highest priority process** first.    * For processes with **equal priority**, use **Round Robin** scheduling.    * **Time quantum** = **minimum remaining burst time** among the ready processes.    * If a process’s remaining time ≤ half of the quantum, execute it fully to reduce overhead. 4. **User Interface (UI):**    * Allow users to **input process details** (AT, BT, Priority).    * Display **Gantt charts**, comparison tables, and graphs of metrics.    * Compute **waiting time, turnaround time, response time, throughput, and context switches** automatically. 5. **Experiments and Testing:**    * Generate **random workloads** with varied arrival times and burst times.    * Evaluate all algorithms on **different datasets**, including larger ones, to address the **research gap**.    * Compare performance metrics and visualize results for analysis. |

## 

## System Architecture (High Level Diagram)(3 pts)

(Provide an overview of the system, identifying its main components and interfaces in the form of a diagram using a tool of your choice).

|  |
| --- |
| 1.**User Input (Process Data):** Arrival time, burst time, priority.  2. **Scheduler Core:** Main engine to handle scheduling.  3. **Algorithms:** Implements FCFS, SJF, Priority, Round Robin, and the proposed Hybrid algorithm.  4. **Visualization:** Generates Gantt charts and statistical summaries.  5. **Output:** Provides comparison and analysis across algorithms.    1. **User Input (Process Data)** → feeds into **Scheduler Core**.  2. **Scheduler Core** → connects to **Algorithms** (FCFS, SJF, Priority, RR, Hybrid).  3. Both Scheduler Core and Algorithms → send results to **Visualization (Gantt Chart, Stats)**.  4. Visualization → passes data to **Performance Metrics** (Waiting, Turnaround, Response, Throughput).  5. Finally → **Output (Comparison & Analysis)** summarizes results. |

## 

## Project Outcome / Deliverables (2 pts)

(Describe what are the outcomes / deliverables of the project. Max 200 words).

|  |
| --- |
| 1. **Interactive Tool for Scheduling Analysis**    * A working system that allows users to **input process details** (arrival time, burst time, priority) and **visualize scheduling results** through Gantt charts, tables, and graphs. 2. **Implementation of Classical Algorithms**    * Correct and tested implementations of **FCFS, SJF (non-preemptive and SRTF), Priority, and Round Robin**, serving as benchmarks for comparison. 3. **Proposed Hybrid Algorithm (MPP / PPRR)**    * Implementation of the **Modified Priority Preemptive Scheduling algorithm**, which combines **Preemptive Priority** with **Round Robin**, using the **shortest burst time in the ready queue as the quantum**.    * Shows **improved fairness**, **reduced waiting time**, and **better response time** compared to classical algorithms. 4. **Performance Analysis Report**    * Comparative evaluation of classical and hybrid algorithms based on **average waiting time, turnaround time, response time, throughput, and fairness**.    * Demonstrates the **advantages of the proposed MPP algorithm** in realistic scenarios with varied arrival times and larger datasets. 5. **Research Contribution**    * Provides a simple yet effective method to **dynamically adapt the RR quantum within a preemptive priority framework**,    * Addresses **starvation** issues and **enhances CPU scheduling efficiency**, filling the research gap left by previous studies. |

# Assumptions

|  |
| --- |
| 1. The CPU is **single-core**, executing only one process at a time.  2. All processes are **independent** (no inter-process communication or I/O blocking considered).  3. Burst times, arrival times, and priorities of processes are **known in advance** and provided as input.  4. Context switching overhead is considered **negligible** for simplicity.  5. Priorities are **static** (do not change during execution unless preempted).  6. The system uses **non-real-time scheduling**, focusing only on batch and interactive processes. |

## References

|  |
| --- |
| 1. **Singh, S., & Goyal, A. (2024).** Efficient CPU Scheduling Algorithm for Multilevel Queue. *Heliyon, 10*(8). Elsevier.  Available at: <https://www.sciencedirect.com/science/article/pii/S2405844024059905>  2. **Kumar, N., & Kumar, S. (2020).** A New Hybrid CPU Scheduling Algorithm: SJF with Round Robin. *Procedia Computer Science, 167*, 2362–2371. Elsevier.  Available at: <https://www.sciencedirect.com/science/article/pii/S1877050920300454>  3. **Silberschatz, A., Galvin, P. B., & Gagne, G. (2018).** *Operating System Concepts* (10th ed.). Wiley.  Standard textbook covering FCFS, SJF, Priority, and Round Robin algorithms.  4. **Stallings, W. (2018).** *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.  Detailed explanations of scheduling algorithms and performance analysis. |