**CPU Scheduling Algorithms Project**

**1. Overview**

This project aims to implement several CPU scheduling algorithms used in operating systems and provide a user-friendly frontend interface. The goal is to offer a comprehensive tool for visualizing and understanding different CPU scheduling techniques, including First-Come, First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling and Round Robin (RR) algorithm.

**2. Project Components**

**a. Backend Implementation**

The backend, developed in C++, handles the core logic and calculations for the CPU scheduling algorithms. It includes essential data structures such as `Process` and `Output` to represent processes and their scheduling results.

**Scheduling Algorithms:**

**- First-Come, First-Served (FCFS):** Processes are scheduled based on their arrival times. The `fcfs()` function sorts processes by arrival time and executes them in that order.

**- Shortest Job First (SJF):** Processes with the shortest burst time are prioritized. The `sjf()` function uses a ready queue to select the process with the shortest burst time.

**- Priority Scheduling:** Processes are scheduled according to their priority. The `priority()` function maintains a ready queue and selects the highest priority process for execution.

**- Round Robin (RR):** Each process is given a time quantum for execution. The `rr()` function cycles through the processes, giving each a time slice.

The backend also includes functions to compute metrics like average waiting time, average turnaround time, and CPU utilization. To compile the backend, use:

g++ -o backend/scheduler backend/algorithms/scheduler.cpp

**b. Frontend Development**

The frontend is built using HTML, CSS, and JavaScript, providing a user-friendly interface. Users can input the number of processes, arrival time, burst time, and priority. They can select a scheduling algorithm and view results such as process details, average waiting time, average turnaround time, CPU utilization, and a graphical Gantt chart.

To set up the frontend, navigate to the project directory and run:

npm install

Start the application with:

npm start

**c. Backend-Frontend Integration**

Node.js with C++ addons is used to integrate the backend and frontend. The frontend sends HTTP requests to the server, which executes the C++ code and returns the results for display.

**3. Key Features and Enhancements**

- Algorithm Comparison: Users can compare the performance of different algorithms side-by-side.

- Best Algorithm Recommendation: The tool analyzes the results and recommends the best algorithm based on the lowest average waiting time and turnaround time.

- Interactive Gantt Chart: Users can interact with the Gantt chart to view additional details about each process block.

- Dynamic Priority Adjustment: Users can adjust process priorities dynamically in the Priority Scheduling algorithm and see the immediate impact on scheduling.

**4. Algorithm Efficiency Analysis**

Evaluation of Algorithms:

**First-Come, First-Served (FCFS)**

- Advantages: Simple and straightforward, no process starvation.

- Disadvantages: Non-preemptive, can lead to higher average waiting time.

**Shortest Job First (SJF)**

- Advantages: Optimizes waiting time.

- Disadvantages: Possibility of starvation for longer processes.

**Priority Scheduling**

- Advantages: Ensures important processes are executed first.

- Disadvantages: Lower-priority processes may experience starvation.

**Round Robin (RR)**

- Advantages: Fair time allocation, preemptive nature prevents starvation.

- Disadvantages: Context switching can be costly.

**5. Conclusion and Future Directions**

The project successfully implemented and visualized various CPU scheduling algorithms through an integrated backend and frontend system. Future enhancements could include incorporating additional scheduling algorithms, refining the user interface, and further optimizing the performance of the backend-frontend integration.