**Process Scheduling Simulator with Multiple Scheduling Algorithms**

Overview

The Process Scheduling Simulator is designed to simulate various process scheduling algorithms commonly used in operating systems. It reads process information from input files and uses different scheduling algorithms to manage the execution order of these processes. The main focus is on evaluating and comparing the performance of these algorithms by visualizing the process execution timeline through Gantt charts and calculating performance metrics such as average turnaround time and average waiting time.

Objectives

The Process Scheduling Simulator is developed with several principal objectives in consideration that aim to provide comprehensive insights into the functionality and efficiency of different process scheduling algorithms within an operating system. Its primary goals are as followed:

* Assess the performance of various scheduling algorithms on a set of processes.
* Generate a Gantt chart that depicts the execution schedule.
* Determine average waiting time and average turnaround time.

Process Scheduling

**Definition**

Process scheduling is a fundamental function of an operating system that manages the execution of processes by determining the order in which processes run. It aims to maximize CPU utilization and ensure efficient process management.

**Objectives**

The main purpose of process scheduling include the following:

* Maximizing CPU utilization.
* Minimizing waiting time of the processes.
* Minimizing turnaround time of the processes.
* Distributing CPU time fairly among processes to avoid starvation.

**Types**

Process scheduling is categorized as either of the below:

* **Preemptive Scheduling:**

It is a type of process scheduling where the operating system can interrupt and suspend the currently running process to allocate the CPU to another process based on certain criteria such as process priority, arrival of a higher priority process, or the expiration of a time quantum.

* **Non-Preemptive Scheduling:**

It is a type of process scheduling where once a process starts its CPU burst, it runs to completion without being interrupted. The CPU will only switch to another process when the current process terminates or voluntarily yields control of the CPU.

**Process Scheduling Algorithms**

There are a number of process scheduling algorithms, and each has its own characteristics and implementation technique. Among them are the following:

* First Come First Serve
* Shortest Job First
* Longest Job First
* Longest Remaining Time First
* Shortest Remaining Time First
* Round Robin
* Priority Scheduling
* Multilevel Queue Scheduling
* Multilevel Feedback Queue Scheduling

Key Features

The Process Scheduling Simulator is a complete tool for analyzing and assessing various scheduling algorithms, offering insightful information about their behavior and performance in a controlled environment. It is integrated with many significant features, such as:

**User Input Handling**

* **File Selection:** The simulator allows users to select an input text file (.txt) that contains process information. This file includes details such as process ID, arrival time, priority, CPU burst time, etc.
* **Simple File Format:** The system ensures ease of use by supporting a straightforward text file format for input data.

**Input Data Parsing and Validation**

* **Robust Parsing:** Reads and interprets process information from the input text file accurately.
* **Error Handling:** Validates input data to handle potential errors or inconsistencies, ensuring reliable simulation results.

**Multiple Scheduling Algorithms**

* **First Come First Serve (FCFS):**
* This algorithm executes processes in the order they arrive.
* It supports non-preemptive scheduling.
* It is simple to use but has a higher average waiting time than other algorithms.
* FCFS may cause Convoy effect.
* **Shortest Remaining Time First (SRTF):** 
  + This algorithm selects the process with the shortest CPU burst time next.
  + This scheduling technique is the preemptive version of Shortest Job First (SJF) algorithm.
  + It is better than FCFS since it lowers the average waiting time.
  + SRTF may lead to starvation.
* **Round Robin (RR):**
* This algorithm allots a fixed time slice (quantum) to each process and cycles through them.
* It is the preemptive version of the FCFS scheduling algorithm.
* RR algorithm generally focuses on Time Sharing method.
* It is easy to use and starvation-free since the CPU is equally allocated to all the processes.
* **Priority Scheduling:**
* This algorithm executes processes based on priority levels (taken as user input); higher priority processes are executed first.
* If there is more than one process with an equal priority value, then the scheduling will occur on the basis of the FCFS algorithm.
* Priority Scheduling could be preemptive method or not.
* It may result in starvation problem.
* **Multilevel Feedback Queue Scheduling (MLFQ):**
* This algorithm divides processes into different queues, each with its own scheduling algorithm.
* It allows different processes to move between various queues.
* It is flexible and adaptable to changing workloads.
* MLFQ is the most complex algorithm.

**Performance Evaluation Metrics**

* **Average Waiting Time:** The simulator calculates the average time that processes spend waiting in the ready queue before execution.
* **Average Turnaround Time:** The simulator calculates the average time taken from process arrival to completion.

**Gantt Chart Generation**

* **Execution Visualization:** The simulator generates Gantt charts to provide a graphical representation of the process execution timeline.
* **Context Switches and Idle Times:** The Gantt chart produced by the simulator clearly marks context switches, the start and end times of each process, as well as the period when the CPU is idle.

**User Interface**

* **Interactive and User-Friendly**: The system is designed with a user-friendly interface that allows easy interaction with the simulator.
* **Algorithm Selection**: Users can select which scheduling algorithm to apply to the input data and evaluate its performance.
* **Customization Options**: The simulator allows users to customize simulation parameters such as time quantum for RR scheduling and priority levels for Priority Scheduling.

**,**

Implementation

The process scheduling simulator is implemented in Java and involves several core functionalities such as parsing input files, selecting scheduling algorithms, scheduling processes, and calculating performance metrics. Below are the detailed implementation aspects of the simulator:

**Parsing Input Files**

The simulator allows users to select an input text file (.txt) containing process information. The selected file is then parsed to extract details such as process ID, arrival time, priority, and CPU burst time. The *JFileChooser* component is used to facilitate file selection.

**Scheduling Algorithm Selection**

The simulator supports multiple scheduling algorithms including FCFS, SRTF, RR, Priority Scheduling, and MLFQ. Users can select the desired algorithm from a dropdown menu, and the corresponding scheduling method is invoked based on the selection.

**Scheduling Algorithms Implementation**

Each scheduling algorithm is implemented as a separate class with a method to perform the scheduling.

* **FCFS.java Module:**
* Initializes process list and Gantt chart list.
* Sorts processes by arrival time.
* For each process, calculates start time, completion time, turnaround time, and waiting time. Updates the Gantt chart.
* Computes the average waiting time and turnaround time.
* Writes results to a file and updates the GUI with process details and the Gantt chart.
* **SRTF.java Module:**
* Initializes process list, Gantt chart list, and other necessary variables.
* Processes are organized based on the shortest remaining time.
* Continuously selects the process with the shortest remaining time, update times, preempts if necessary. Updates the Gantt chart.
* Computes the average waiting time and turnaround time.
* Writes results to a file and updates the GUI with process details and the Gantt chart.
* **RoundRobin.java Module:**
* Initializes process list, Gantt chart list, time quantum, and other necessary variables.
* Processes are organized in a circular queue for time-sharing.
* For each time quantum, updates the current process's times, checks for completion, and rotates the queue. Updates the Gantt chart.
* Computes the average waiting time and turnaround time.
* Writes results to a file and updates the GUI with process details and the Gantt chart.
* **PreemptivePriorityScheduling.java Module:**
* Initializes process list, Gantt chart list, and other necessary variables.
* Processes are organized based on priority taken from the user.
* Continuously selects the process with the highest priority, updates times, preempts if necessary. Updates the Gantt chart.
* Computes the average waiting time and turnaround time.
* Writes results to a file and updates the GUI with process details and the Gantt chart.
* **MLFQ.java Module:**
* Initializes process list, Gantt chart list, multiple queues for different priority levels, and other necessary variables.
* Processes are placed into different queues based on their behavior and CPU burst characteristics.
* Selects processes from queues based on the scheduling policy, updates times, and moves processes between queues as necessary. Updates the Gantt chart.
* Computes the average waiting time and turnaround time.
* Writes results to a file and updates the GUI with process details and the Gantt chart.

**Performance Metrics Comparison**

The simulator uses the Comparison.java module to manage comparison of scheduling algorithm results and display them using graphical representation and ComparisonChart.java module to visualize and display a comparison of scheduling algorithm results using bar charts.

**Gantt Chart Generation**

The simulator generates Gantt charts to visually represent the execution timeline of processes for each scheduling algorithm. This involves marking the start and end times of each process and highlighting any context switches.

**Error Handling and Validation**

The simulator includes error handling mechanisms to ensure robustness, such as validating the input file for potential errors or inconsistencies.

By implementing these functionalities, the process scheduling simulator provides a comprehensive tool for evaluating and comparing different scheduling algorithms.

Usage of Process Scheduling Simulator

The scheduling simulator is a powerful tool for understanding and analyzing different CPU scheduling strategies and has numerous real-world applications. Its ability to simulate diverse scenarios and provide quantitative metrics makes it an invaluable and a versatile tool that can be used across academia, research, and industry to study, analyze, and optimize CPU scheduling strategies and other purposes. Following points demonstrate its significance across various domains:

**Algorithm Comparison:**

One of the primary uses of a scheduling simulator is to compare the performance of different scheduling algorithms under various workloads. Users can input different sets of processes with varying burst times, priorities, and arrival times to observe how each algorithm behaves. Moreover, the simulator outputs metrics such as average turnaround time and average waiting time, which allow users to quantitatively assess which algorithm performs best under specific conditions.

**Educational Tool:**

For students and researchers, scheduling simulators provide a practical way to understand the intricacies of each scheduling algorithm. It also offers visual aid such as Gantt charts that make it easier to comprehend how processes are scheduled over time and how different algorithms impact the overall system performance.

**System Design and Optimization:**

In system design and optimization, simulators can play a crucial role in predicting the behavior of proposed scheduling strategies before implementation. This helps in making informed decisions about system design, resource allocation, and performance tuning.

**Algorithm Development and Testing:**

Before deploying new scheduling algorithms in production systems, simulators provide a controlled environment for testing and validating their effectiveness. In addition, they allow testing of edge cases and extreme scenarios that may be difficult to reproduce in a real-time system. This ensures that algorithms handle rare but critical situations effectively.

**Research and Experimentation:**

Researchers can use scheduling simulators to test hypotheses and explore new scheduling paradigms. By modifying existing algorithms or proposing novel approaches, researchers can advance the field of operating system scheduling. Furthermore, simulation results can be used in academic publications to substantiate findings and contribute to the body of knowledge in operating system design and scheduling.

Performance Metrics of Scheduling Algorithms

Each CPU scheduling algorithm in an operating system is evaluated based on several performance metrics that quantify how efficiently the system handles processes. The Process Scheduling Simulator under discussion uses following metrics to evaluate and compare the performance to algorithms:

**Average Turnaround Time:**

The average time taken for a process to complete execution, including both waiting time and execution time. A scheduling algorithm that regularly produces shorter turnaround times for processes is considered more efficient than one with longer turnaround times.

Turnaround Time = Completion Time – Arrival Time

Average Turnaround Time =

Following is how various algorithms influence the average turnaround time:

* **First Come First Served Algorithm:**

It is influenced by the order of process arrival, but is often larger than other algorithms.

* **Shortest Remaining Time First Algorithm:**

SRTF also tends to have a lower average turnaround time compared to other algorithms due to its ability to prioritize shorter jobs.

* **Round Robin Algorithm:**

It can vary based on the time quantum and the mix of process burst times. Shorter time quantum values typically lead to lower turnaround times but higher context switch overhead.

* **Preemptive Priority Scheduling Algorithm:**

Processes with higher priorities typically have lower turnaround times since they are scheduled more frequently.

* **Multilevel Feedback Queue Algorithm:**

MLFQ optimizes turnaround time by allowing shorter jobs to finish quickly in higher priority queues before longer jobs.

**Average Waiting Time:**

The average time a process spends waiting in the ready queue before it starts execution. A scheduling method that consistently results in reduced wait times for processes is considered more efficient than one that regularly results in longer wait times.

Waiting Time = Turnaround Time – Burst Time

Average Waiting Time =

Below is how different algorithms impact the average waiting time:

* **First Come First Served Algorithm:**

Since FCFS is non-preemptive, the waiting time for each process depends on the arrival times of other processes, but it is generally quite high.

* **Shortest Remaining Time First Algorithm:**

SRTF typically results in the lowest average waiting time among all scheduling algorithms. This is because it always selects the process with the shortest remaining burst time, minimizing waiting times for shorter processes.

* **Round Robin Algorithm:**

Round Robin aims to provide fair scheduling by allowing each process to run for a fixed time quantum. The average waiting time can be influenced by the time quantum: longer quantum times generally result in higher waiting times.

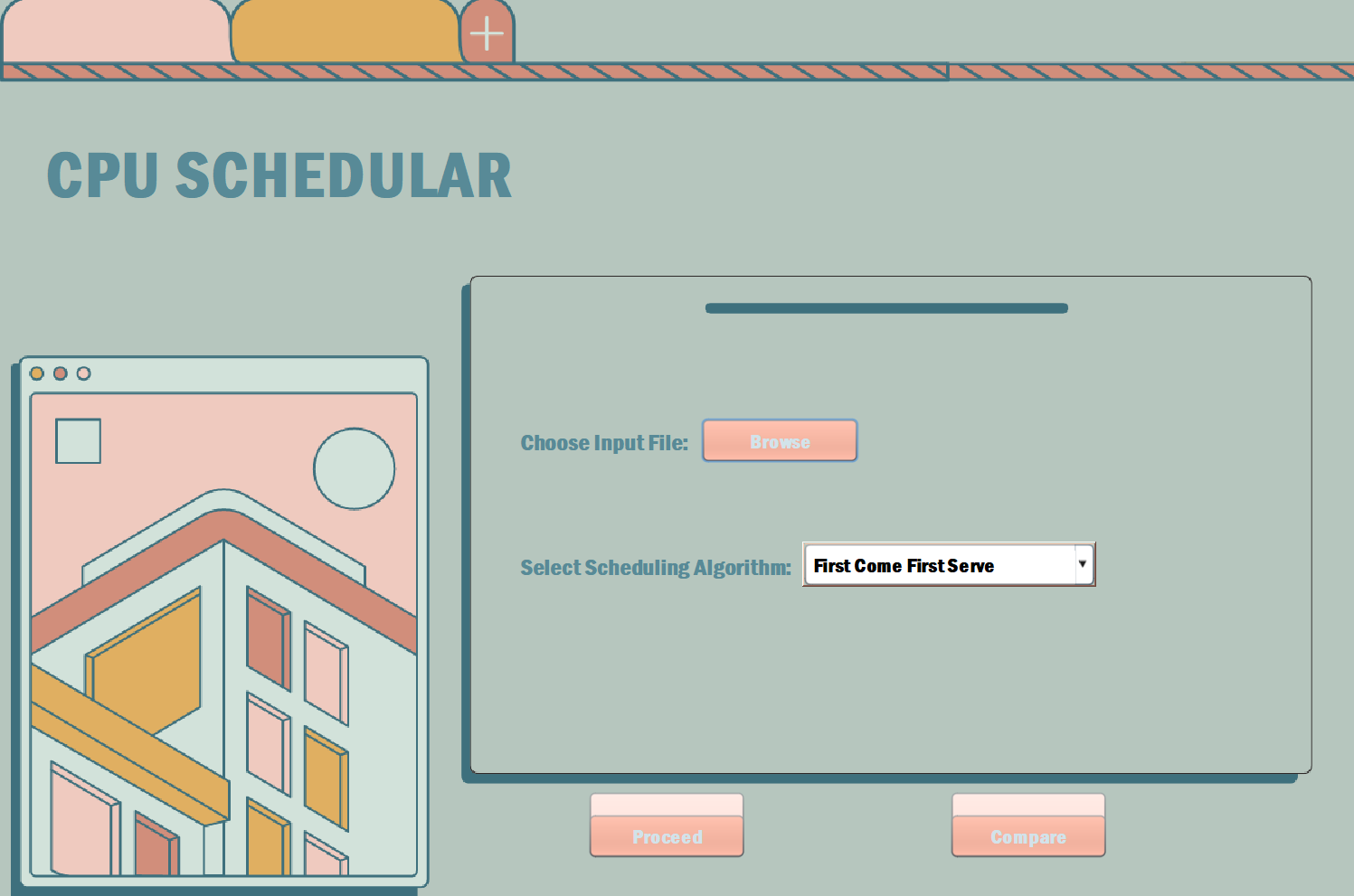
* **Priority Scheduling Algorithm:**

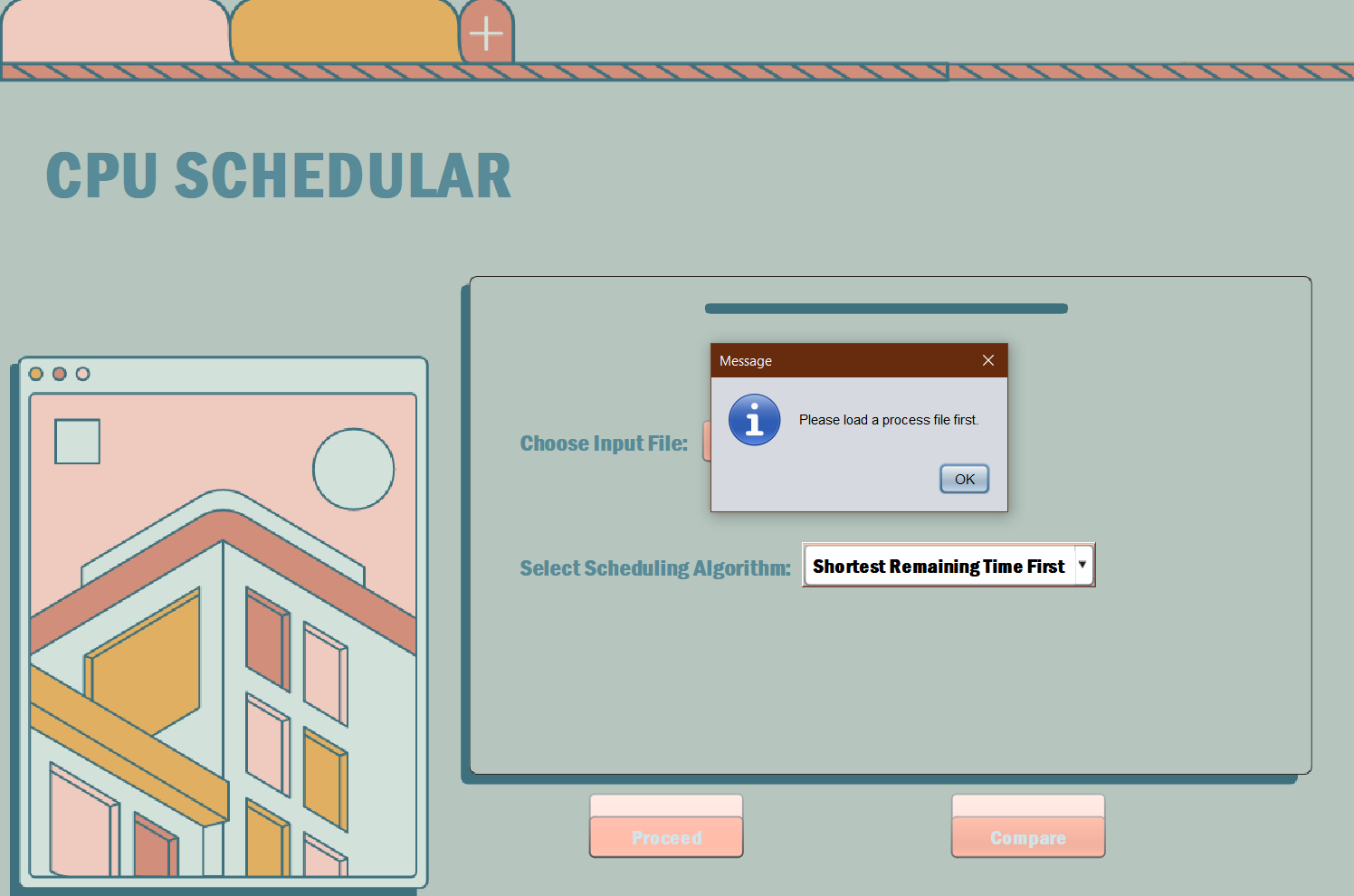
Depends on the priority assignment; higher priority processes generally have lower waiting times. Lower priority processes may suffer from longer waiting times or even starvation if higher priority processes continually arrive.

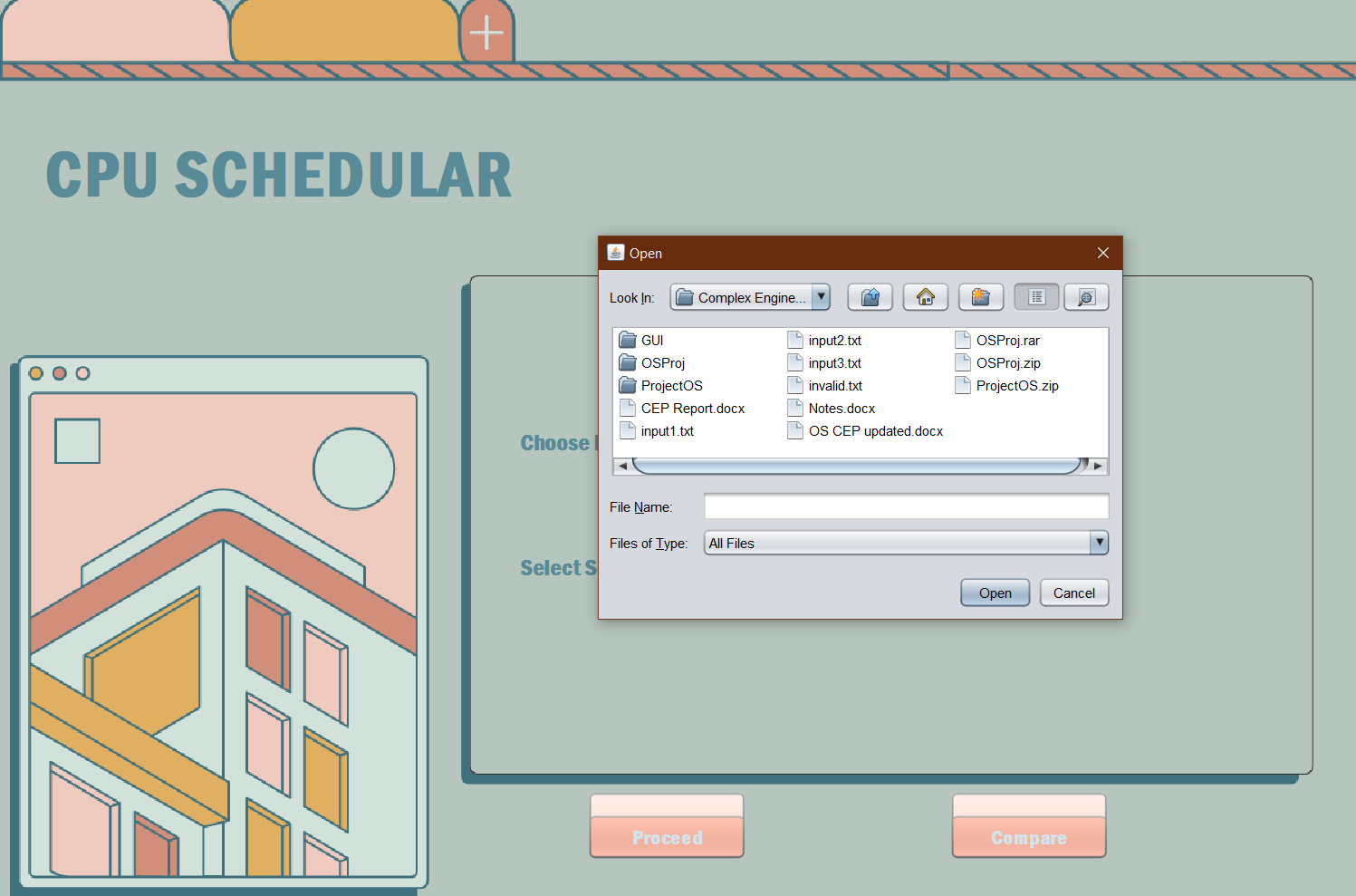
* **Multilevel Feedback Queue Algorithm:**

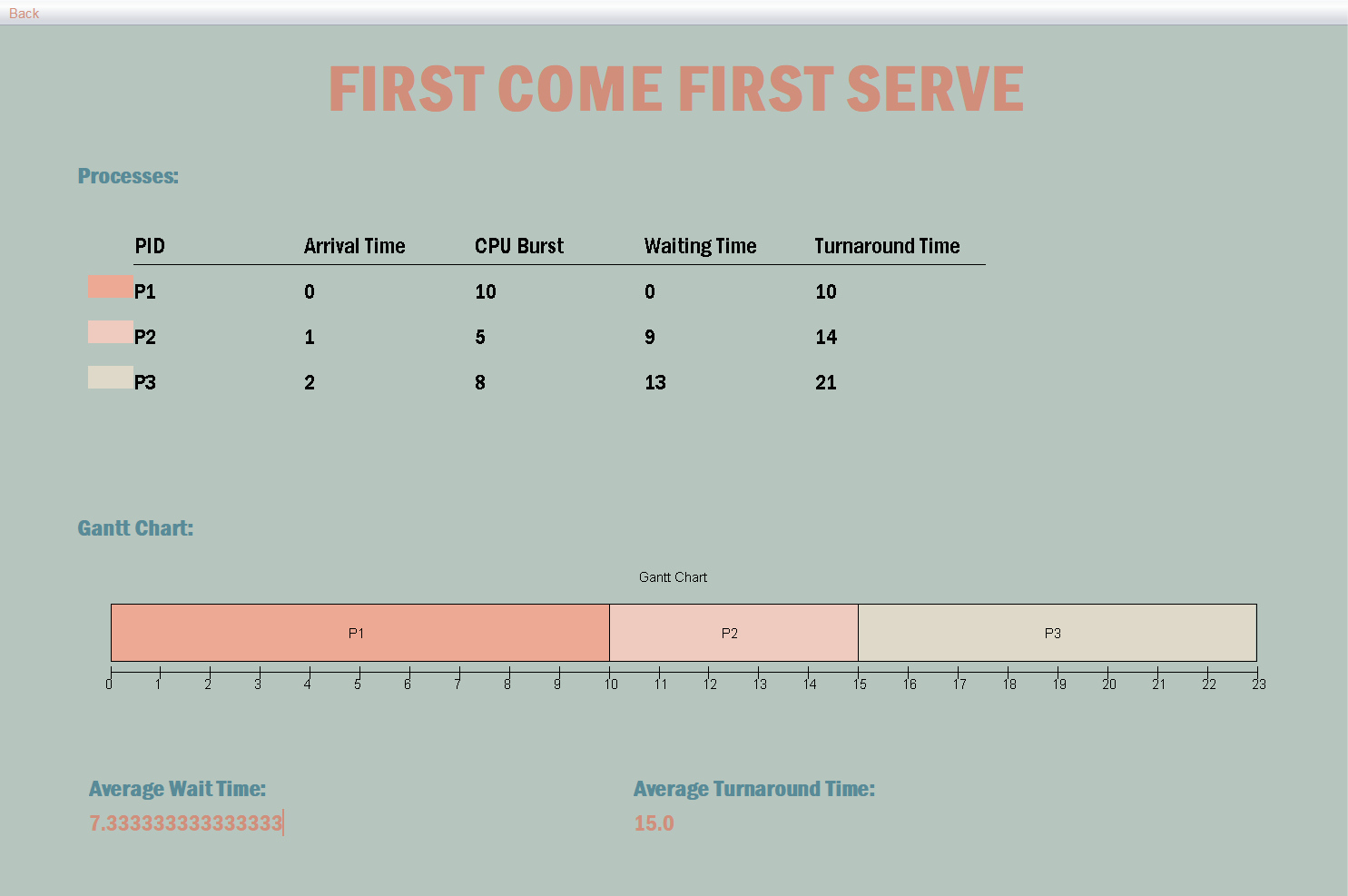
MLFQ aims to minimize waiting times by dynamically adjusting process priorities based on their behavior. Shorter jobs are typically prioritized, leading to reduced waiting times.

In addition to these two, there are other metrics as well through which we can gauge the performance of a scheduling algorithm. Among them are CPU utilization, throughput, response time, and more. These metrics provide valuable insights into how different algorithms behave under varying conditions, aiding in the selection and optimization of scheduling strategies in operating systems.

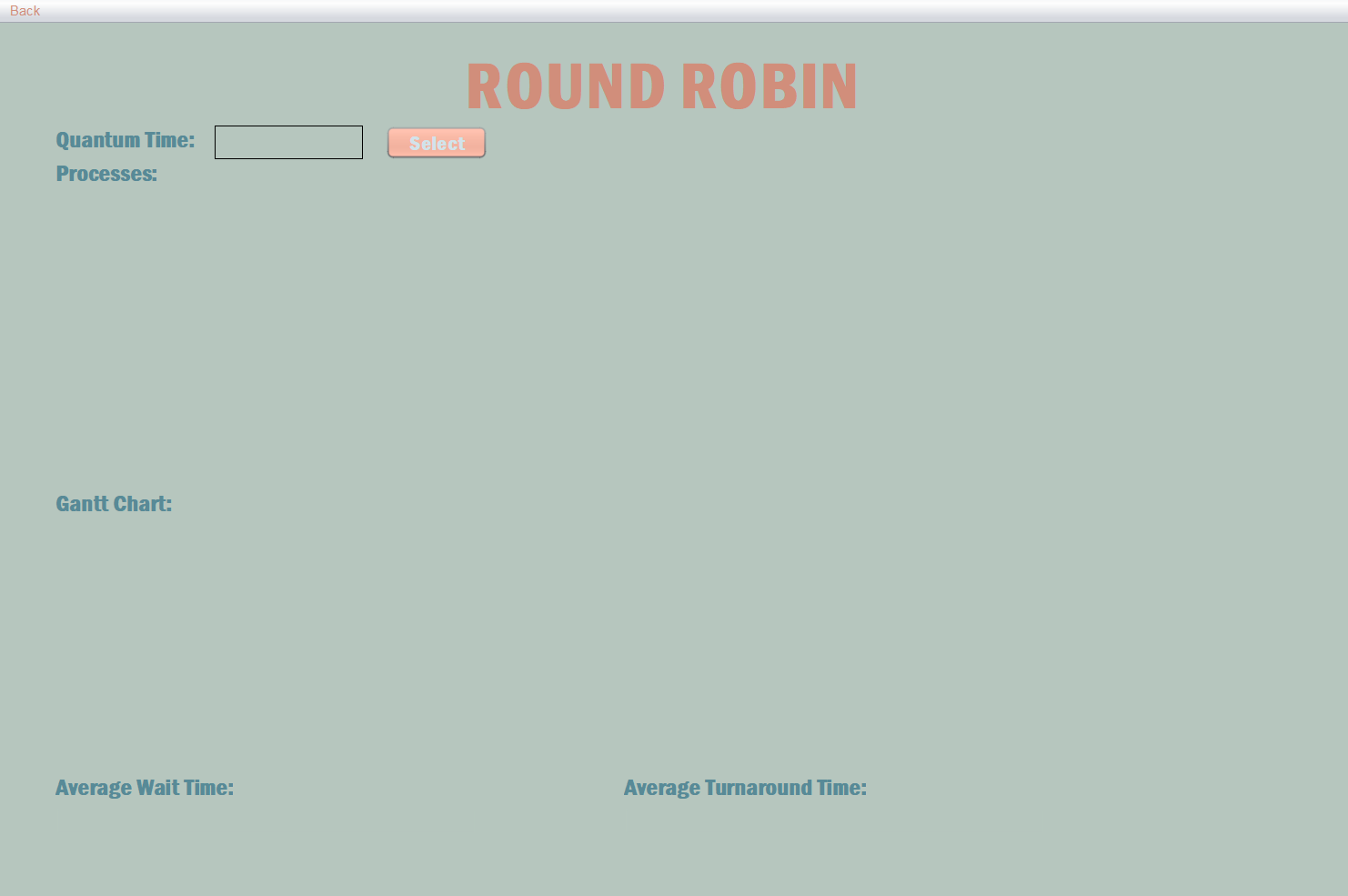
Graphical User Interface

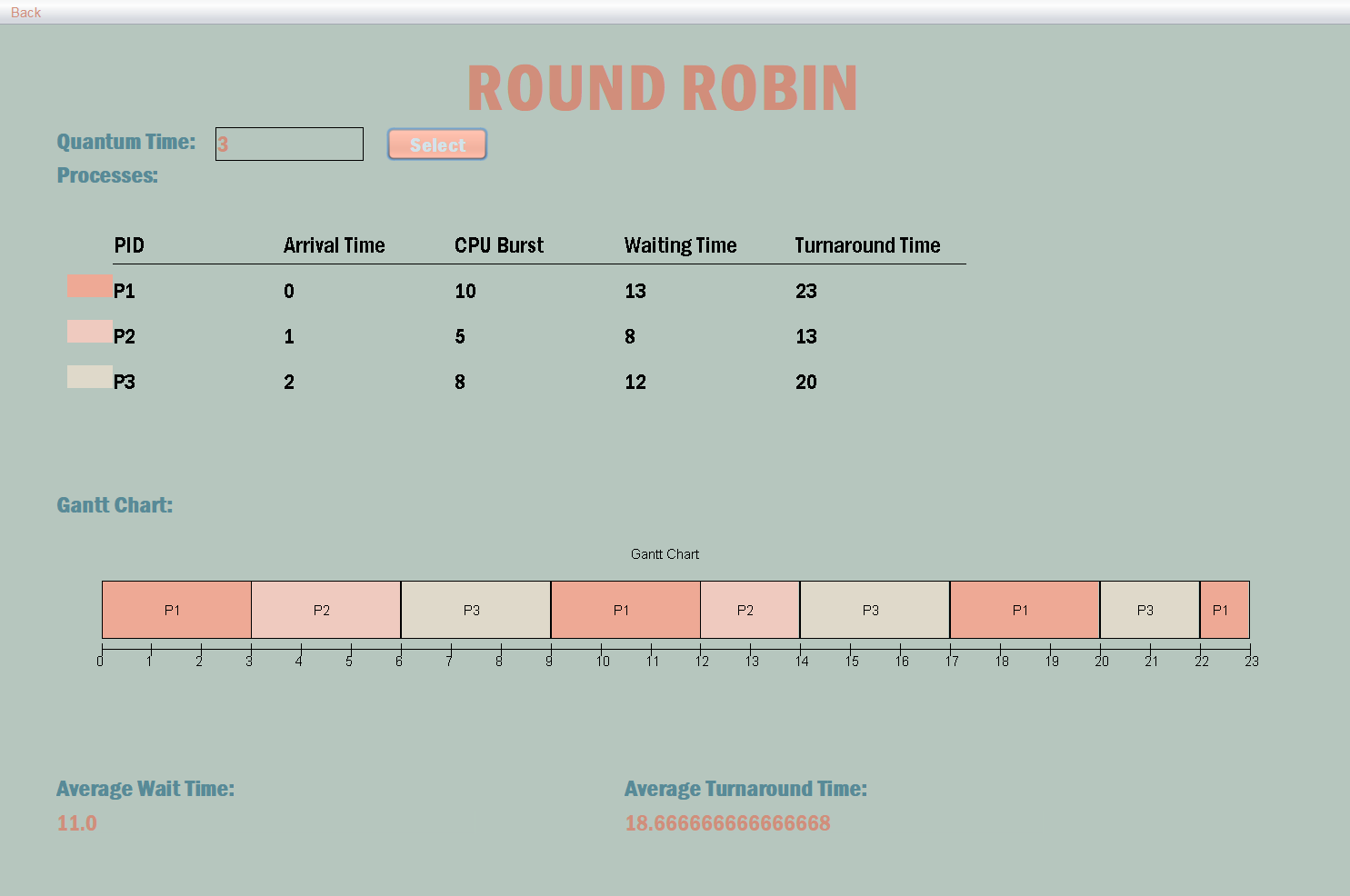


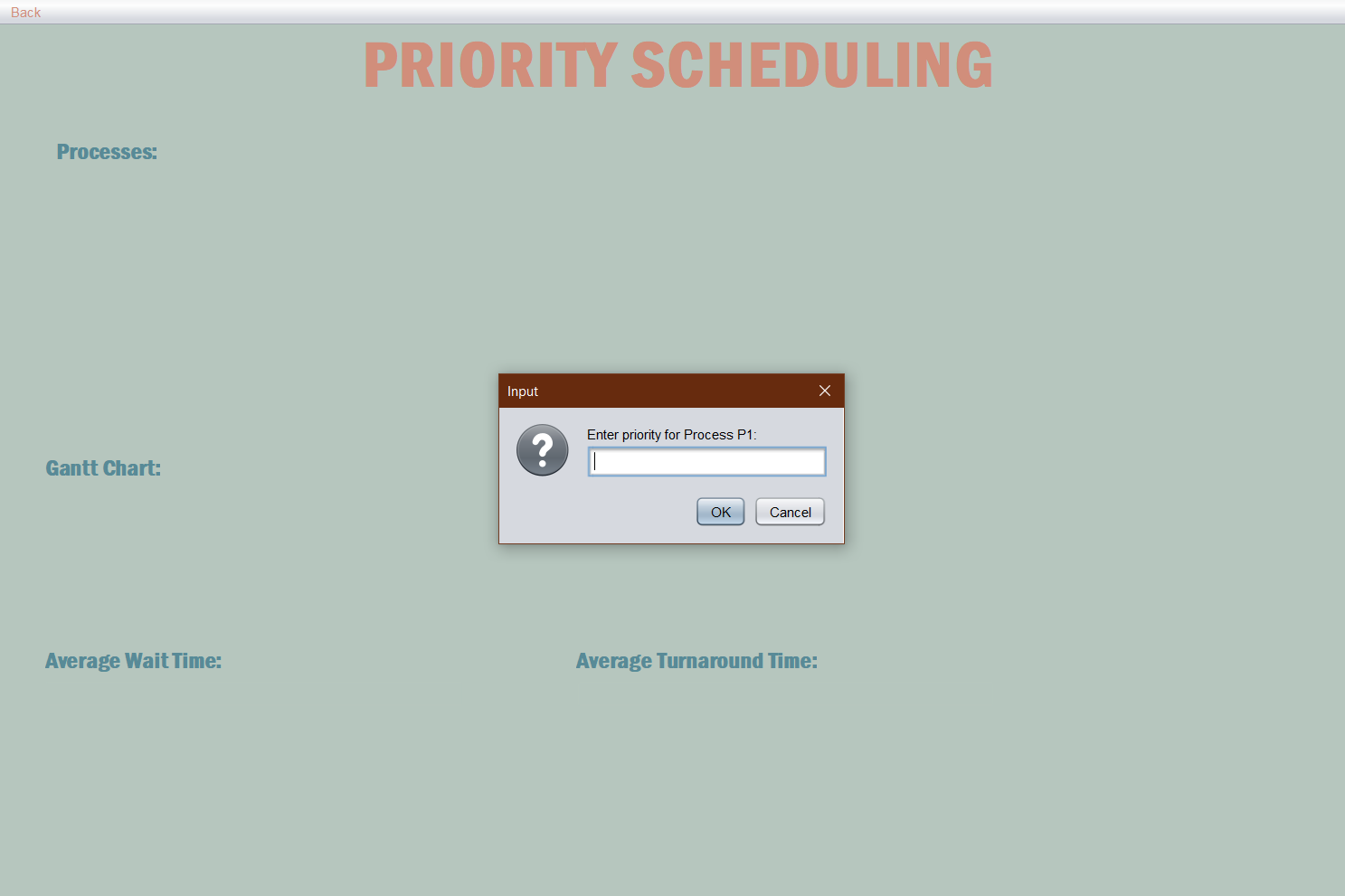




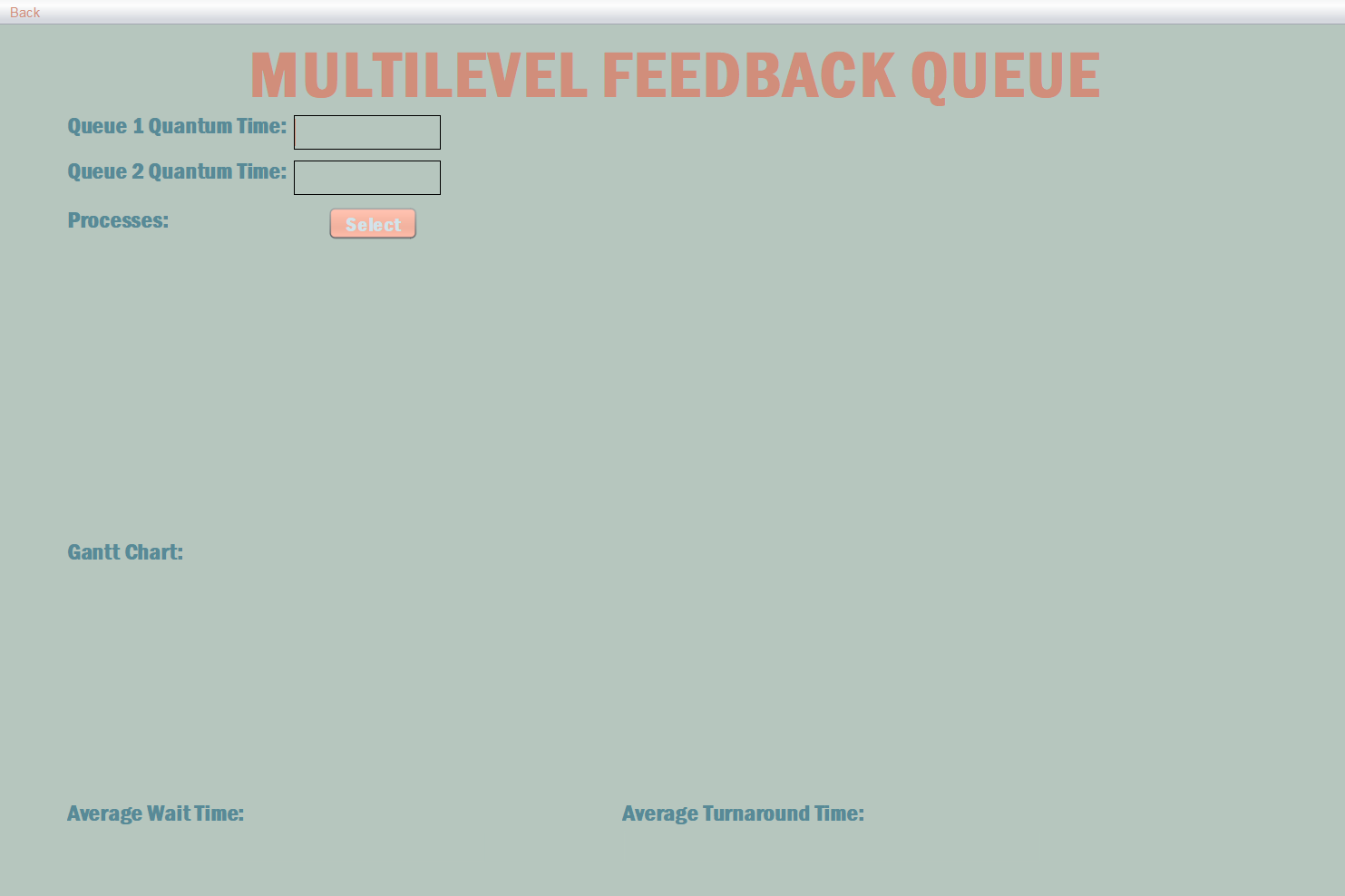


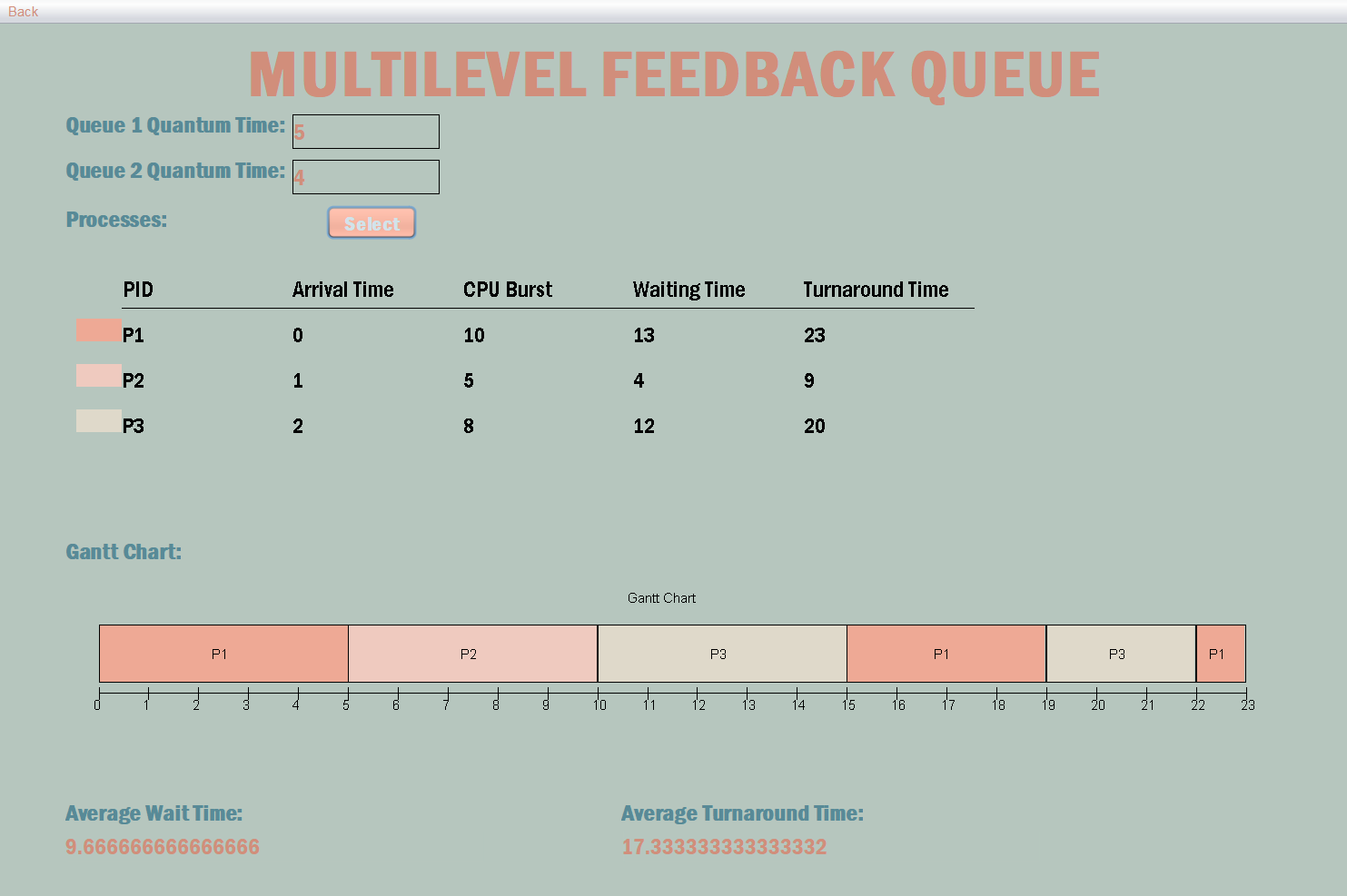


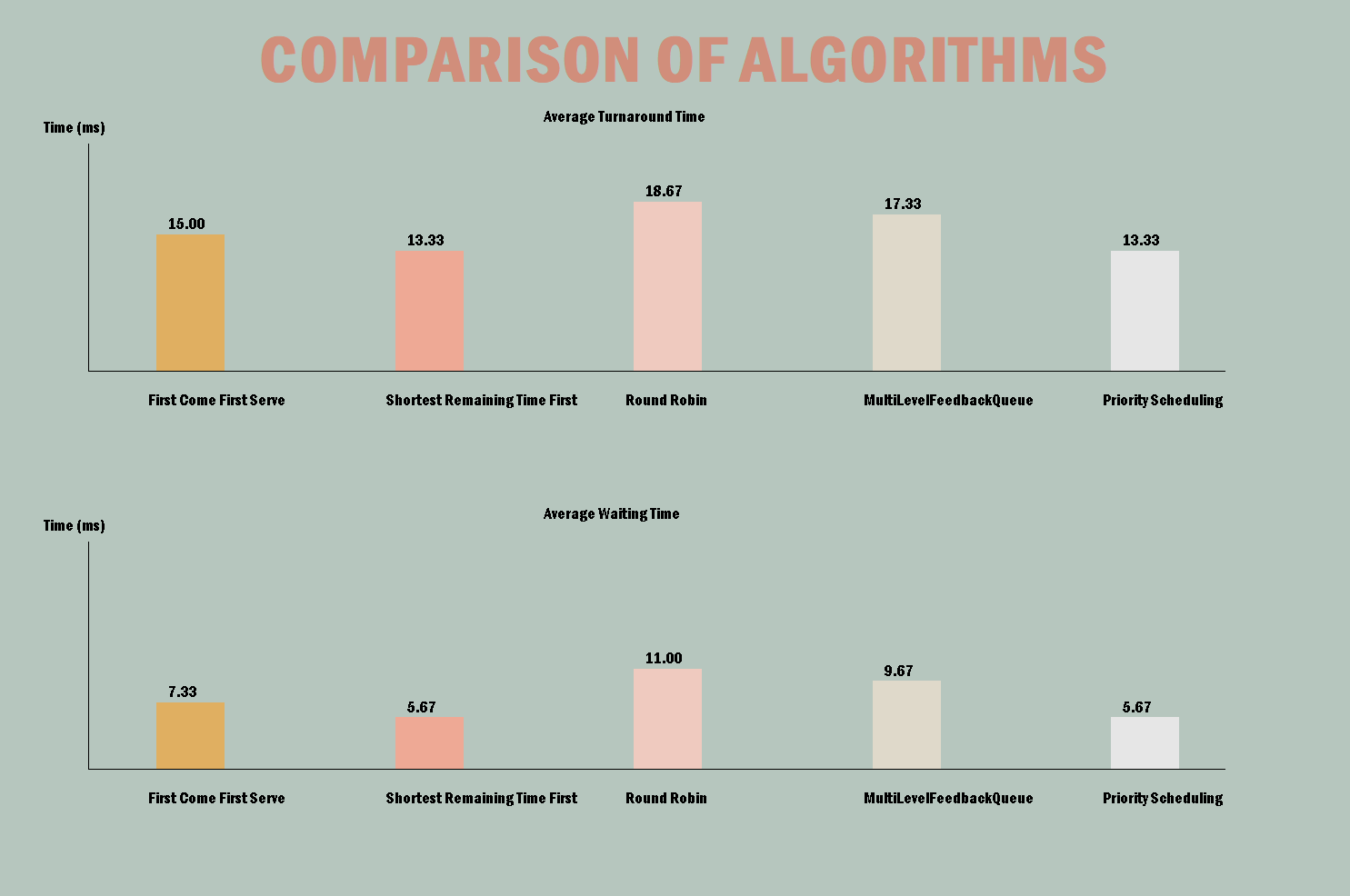












Conclusion

The Process Scheduling Simulator is a robust and versatile Java application capable of simulating diverse scheduling algorithms such as First Come First Served (FCFS), Shortest Remaining Time First (SRTF), Round Robin (RR), Multilevel Feedback Queue (MLFQ), and Priority Scheduling. Each algorithm is meticulously implemented to showcase its unique characteristics in handling process execution, CPU allocation, and performance metrics like average waiting and turnaround times. Moreover, through intuitive graphical representations, including Gantt charts and bar charts, the simulator offers visual depiction of the execution timeline and comparative performance metrics of different algorithms. In essence, the Process Scheduling Simulator not only enhances understanding of fundamental scheduling algorithms but also provides a practical tool for students, researchers, and professionals to explore and compare performance of multiple scheduling algorithms under varied scenarios.