Comprehensive Report for CPU Scheduling Simulation Project

Mohammed ALDiri 443102724

Khaled ALBaker 443100645

Omar ALJebreen 443101949

Under the supervision:

Dr.Mejdl Safran

1. Software and Hardware Tools Used

Software Tools

- **Java Development Kit (JDK)**: Version 17 was used for compiling and running the simulation.
- **Integrated Development Environment (IDE)**: IntelliJ IDEA for project development and debugging.
- **Apache POI Library**: For generating Excel reports.
- **Operating System**: Windows 11(64-bit), macOS Sonoma

Hardware Tools

- **Processor**: 13th Gen Intel(R) Core(TM) i7-1360P,Apple M2 .
- **Memory**: 16 GB DDR4 RAM , 8GB DDR4 RAM.
- **Storage**: 1024 GB SSD, 512 GB SSD.

2.System Calls

The system incorporates several system calls to manage processes and memory effectively. Below is a detailed description:

2.1 startProcess(Job job)

- **Arguments**: A `Job` object containing job details.
- **Expected Output**: Updates the job state to `RUNNING` and logs the start of execution.
- **Functionality**: Simulates the transition of a job from the `READY` to `RUNNING` state.

2.2 terminateProcess(Job job)

- **Arguments**: A `Job` object containing job details.
- **Expected Output**: Updates the job state to `TERMINATED` and releases allocated memory.
- **Functionality**: Simulates job completion and memory deallocation.

2.3 getProcessInfo(Job job)

- **Arguments**: A `Job` object.
- **Expected Output**: Returns job details including ID, burst time, memory requirements.
- **Functionality**: Print the details of job including ID, burst time, memory requirement.

2.4 allocateMemory(Job job)

- **Arguments**: A `Job` object containing memory requirements.
- **Expected Output**: Allocates memory if available; otherwise, logs insufficient memory.
- **Functionality**: Check if there is a space in memory for the job if true allocate memory and return true otherwise false.

2.5 releaseMemory(Job job)

- **Arguments**: A `Job` object.
- **Expected Output**: Decreases used memory by the job's requirement.
- **Functionality**: remove the job memory required from the total memory.

3. Analysis of Program Strengths and Weaknesses

Strengths

- **Multithreading Support**: Efficient use of threads for job loading, memory management, and scheduling.
- **Flexible Scheduling**: Implements multiple scheduling algorithms: FCFS, Round Robin, and SJF.
- **Scalability**: Easily extensible to include more scheduling algorithms or system functionalities.
- **Detailed Reporting**: Outputs performance metrics and generates Excel reports using Apache POI.

Weaknesses

- **Limited Memory Handling**: Fixed memory size of 1024 MB may restrict simulation with larger datasets.
- **Lack of Preemptive Scheduling**: Current implementation lacks advanced preemptive algorithms like Priority Scheduling.
- **Assumptions in Job Characteristics:** Certain assumptions, such as jobs arriving with memory requirements predefined or burst times known in advance, may not align with more dynamic or realistic CPU scheduling scenarios.

4. Threads Created and Multithreading Impact

Threads Created

- **JobLoader Thread**: Reads and parses jobs from a file into the job queue.
- **MemoryManager Thread**: Monitors memory availability and moves jobs to the ready queue.
- **Scheduler Thread**: Executes the selected scheduling algorithm.

Multithreading Impact on Performance

- **Improved Efficiency**: Multithreading ensures concurrent execution of job loading, memory allocation, and scheduling.
- **Reduction in Idle Time**: ReadyQueue management in parallel with job processing reduces CPU idle time.
- **Potential Bottleneck**: Multiple threads and inefficient thread synchronization might cause delays.

5. Comparison of Output Formats

Gantt Charts

- **Advantages**:
    - Visual representation of job execution timelines.
    - Easier to analyze overlaps and idle periods.
- **Disadvantages**:
    - Limited clarity with a high number of jobs.

Tables

- **Advantages**:
    - Detailed numerical data (e.g., waiting and turnaround times).
    - Easier to calculate averages and compare metrics.
- **Disadvantages**:
    - Lacks visual appeal.

Preference

Gantt charts are preferred for their clarity in visualizing execution order and timing. However, tables complement them by providing detailed metrics for deeper analysis.

6. Proposal for Simulating an Entire Operating System

To expand the program to simulate an entire operating system, the following modifications are proposed:

6.1 Process Management

- **Enhancements**:
    - Implement process creation and termination mechanisms.

    - Introduce inter-process communication (IPC).

6.2 File System Simulation

- **Implementation**:
    - Design a virtual file system with operations like read, write, and delete.
    - Maintain file metadata (e.g., size, creation date).

6.3 Device Management

- **Features**:
    - Simulate device drivers for input/output management.
    - Implement device queues for handling I/O requests.

6.4 Memory Management

- **Improvements**:
    - Add virtual memory management with paging and segmentation.
    - Implement dynamic memory allocation and deallocation.

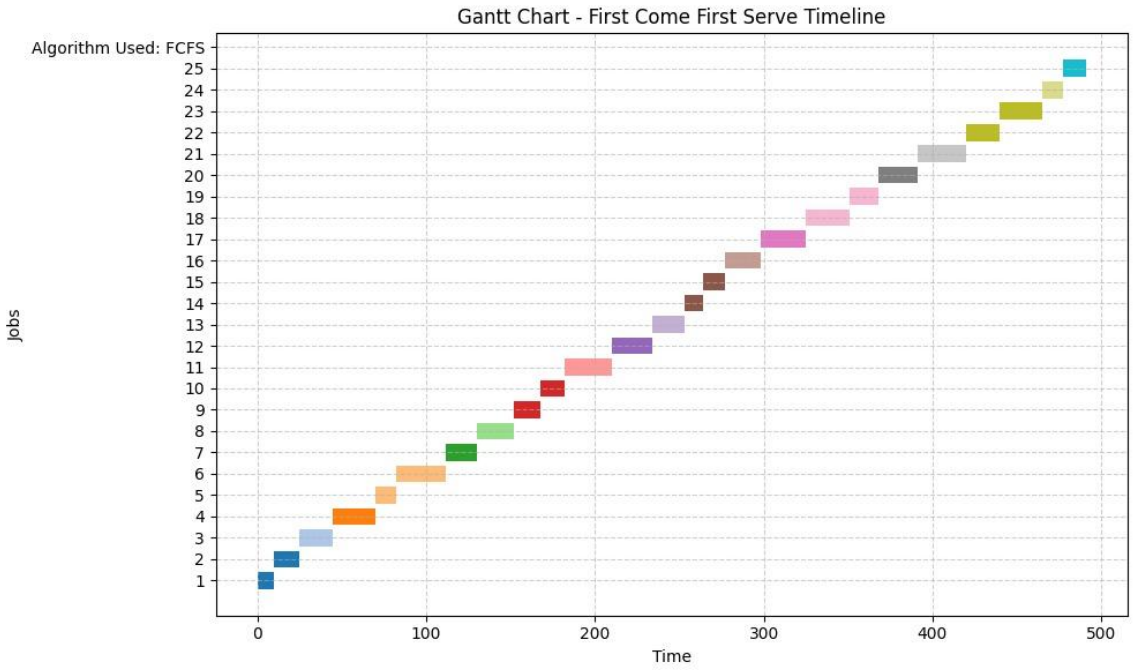
6.5 Scheduling Enhancements

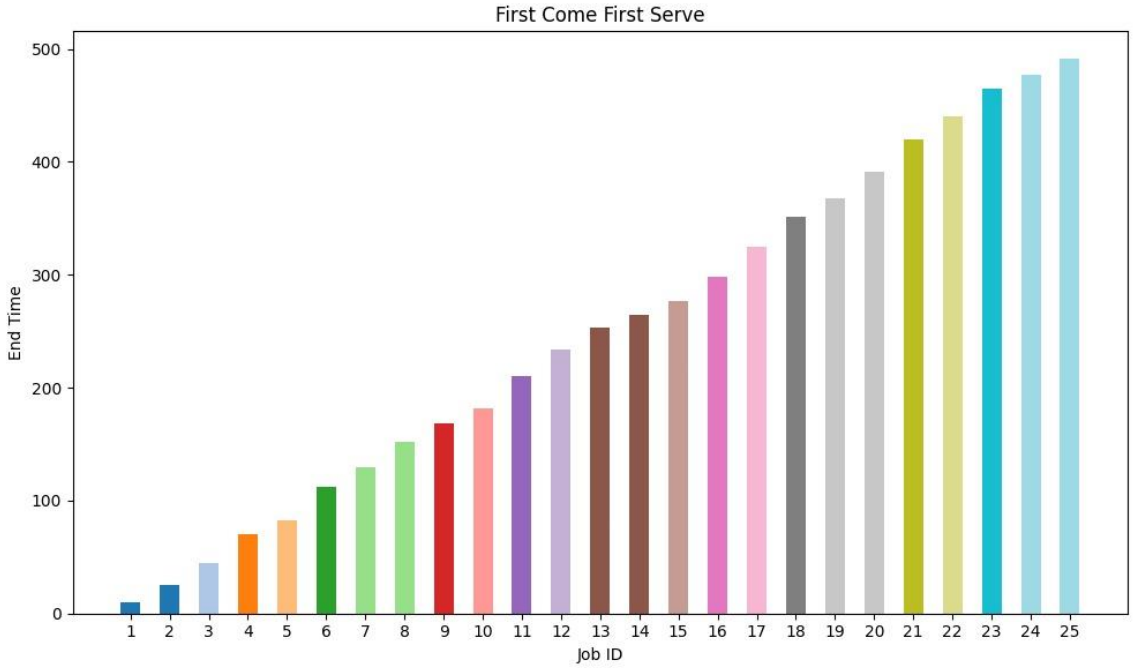
- **Future Algorithms**:
    - Priority Scheduling (preemptive and non-preemptive).
    - Multi-level Queue Scheduling.

7.Performance Figures

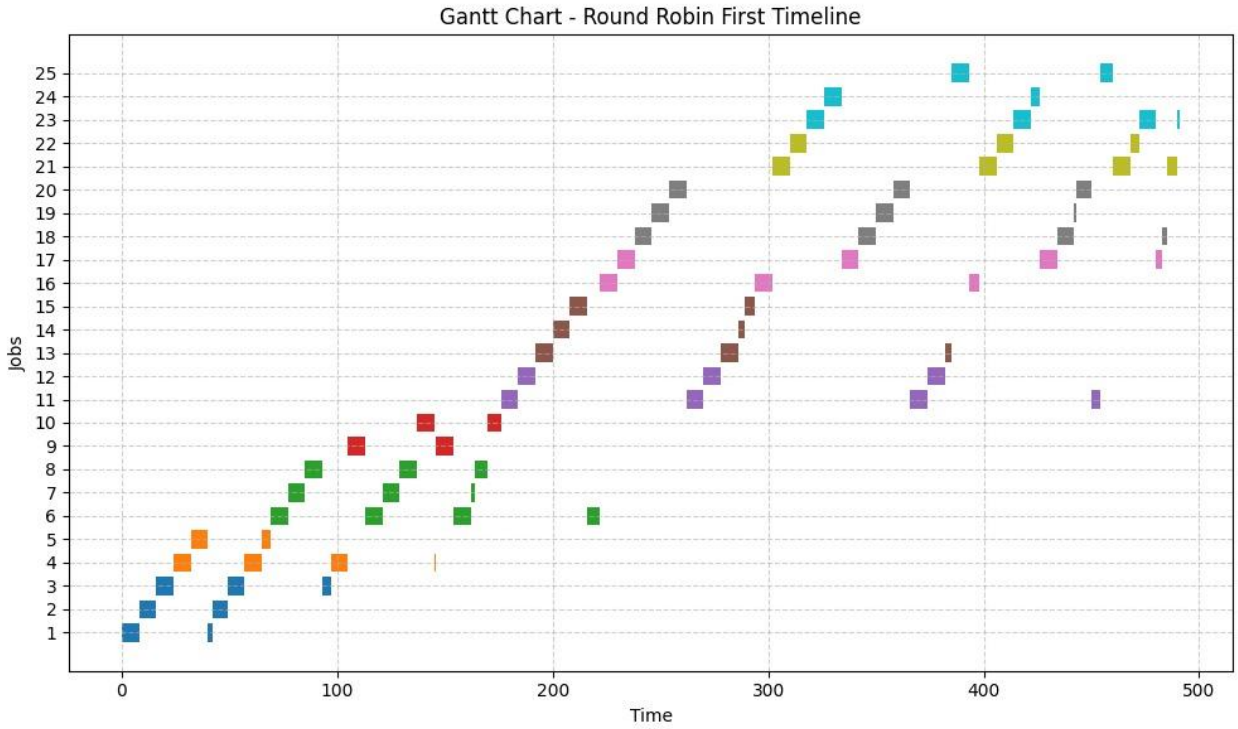
- **The following figures shall illustrate the behavior and performance for each algorithm**
- **Figure 7 shall show the base comparison between all three algorithms**

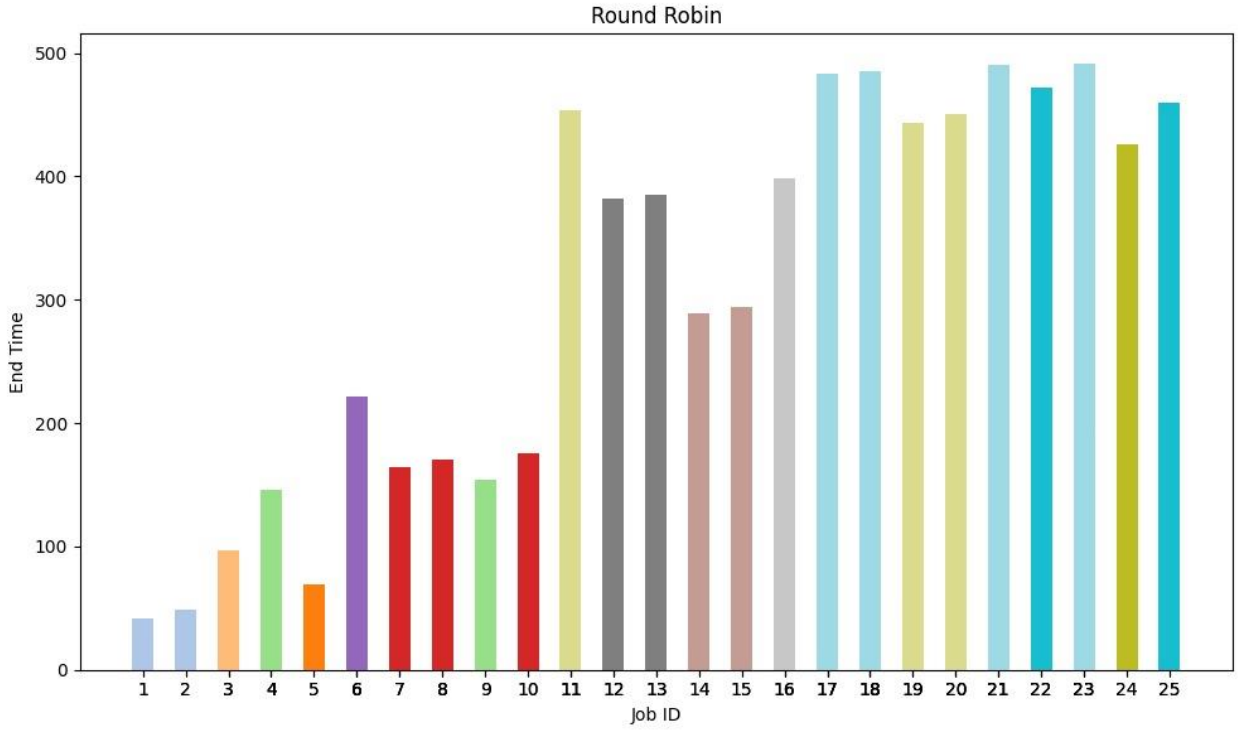
First Come First Serve



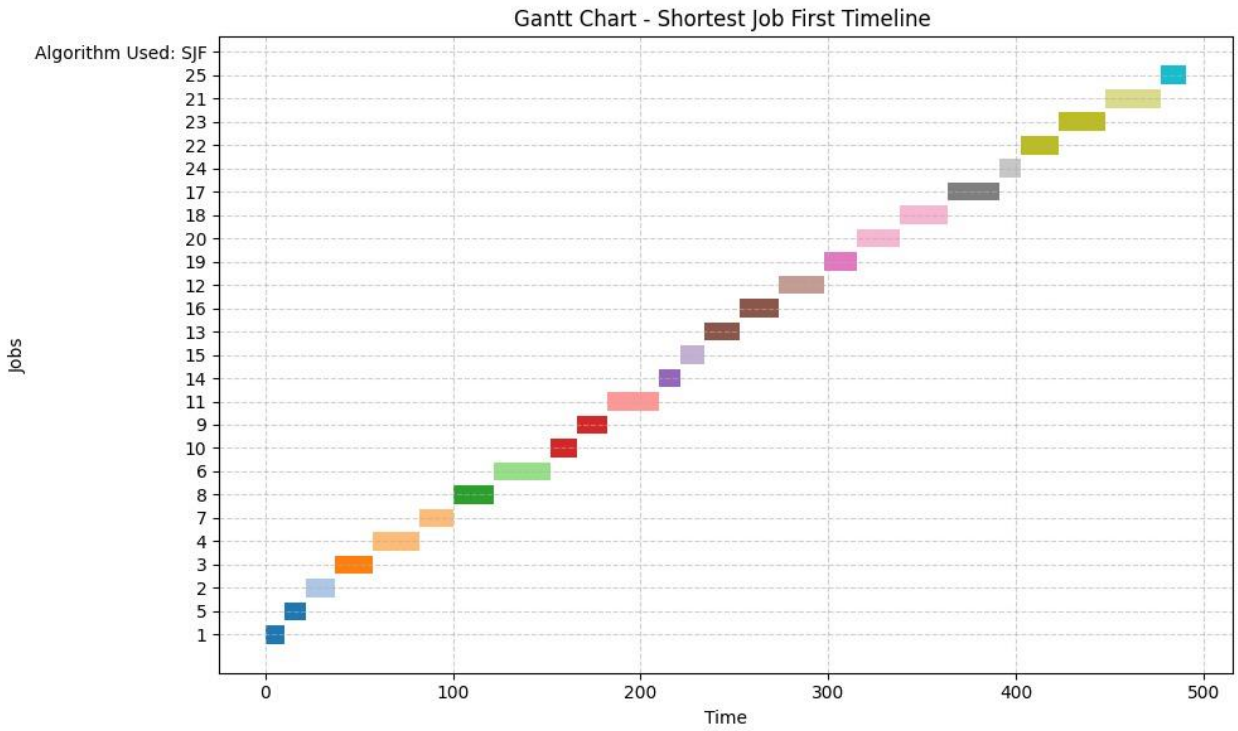


Round Robin


Gantt Chart - Round Robin First Timeline
Jobs
Time
0
100
200
300
400
500


Round Robin
End Time
Job ID
0
100
200
300
400
500
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Shortest job first



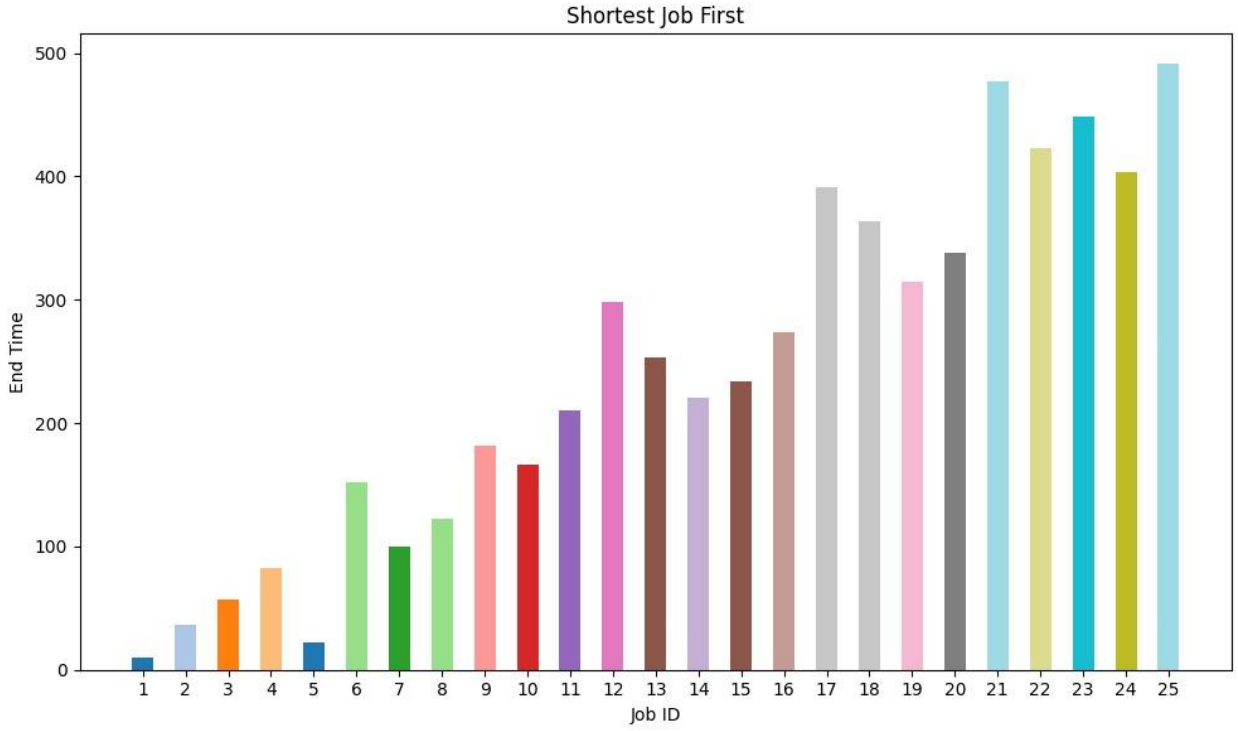
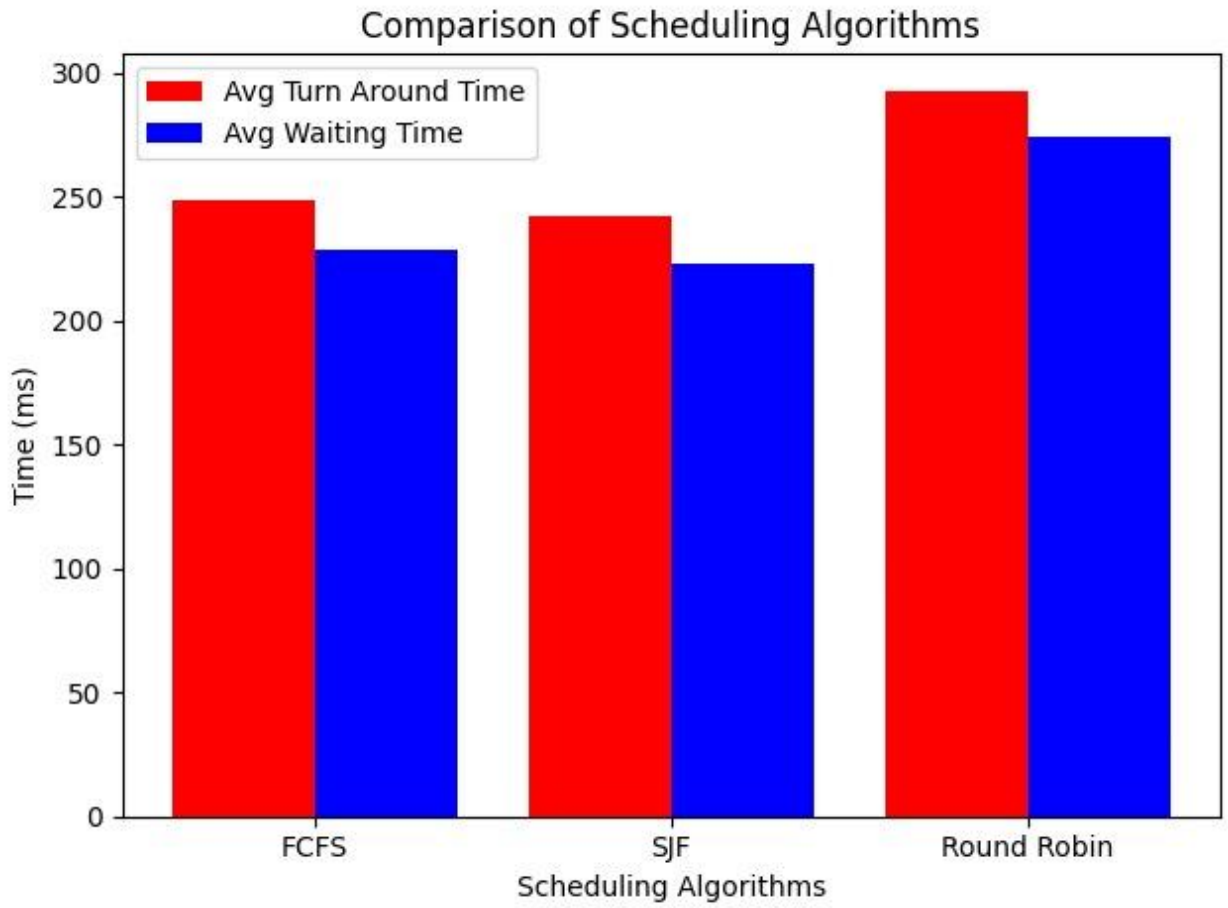


Figure 7


Comparison of Scheduling Algorithms
Avg Turn Around Time
Avg Waiting Time
Time (ms)
300
250
200
150
100
50
0
FCFS
SJF
Round Robin
Scheduling Algorithms