

**Course Code:** CSE325

**Course Title:** Operating System

**Project Report:** CPU Scheduling Simulator

**Submitted By:**

**Name:** Humaira Habib Usha

**Id:** 2023-1-60-022

**Sec:** 04

**Submitted To:**

Prof. Dr. Md. Motaharul Islam

Adjunct Faculty

Department of CSE

**Submission Date:**

26.08.2025

**CPU Scheduling Algorithm Simulator**

**Overview**

This project is a Windows-based tool that shows how CPU scheduling algorithms work in operating systems. It lets students and teachers see process execution through Gantt charts, compare different algorithms, and learn scheduling concepts in an interactive way.

**Features**

Core Features:

- Multiple Scheduling Algorithms: FCFS, SJF, Round Robin, Priority Scheduling

- Interactive GUI: Native Windows interface built with Windows API

- Real-time Visualization: Dynamic Gantt chart generation

- Performance Metrics: Automatic calculation of waiting times, turnaround times

- Algorithm Comparison: Side-by-side performance analysis

- Process Management: Add, modify, and remove processes dynamically

**Technical Features:**

- Zero External Dependencies: Uses only Windows API and standard C++ libraries

- Memory Efficient: Optimized data structures for large process sets

- Cross-Build Support: Both Debug and Release configurations

- Portable: Self-contained executable with static linking

**Supported Algorithms**

1. First Come First Serve (FCFS)

Type: Non-preemptive

Strategy: Processes executed in order of arrival

2. Shortest Job First (SJF)

Type: Non-preemptive

Strategy: Selects process with shortest burst time

3. Round Robin (RR)

Type: Preemptive

Strategy: Time quantum-based circular scheduling

4. Priority Scheduling

Type: Non-preemptive

Strategy: Processes scheduled based on priority values

**Usage Guide**

Adding Processes:

- Launch application

- Input arrival time, burst time, priority

- Click "Add Process"

Running Algorithms:

- Select FCFS, SJF, RR, or Priority

- Set time quantum for RR

- Execute and view Gantt chart

Comparing Algorithms:

- Add multiple processes

- Run 'Compare All Algorithms'

- Analyze results

**Technical Implementation**

Core Data Structures: struct Process, struct GanttEntry

Algorithm Pattern: CPUScheduler class with static methods

Performance Metrics:

- avgWaitingTime = sum(waitingTime) / n

- avgTurnaroundTime = sum(turnaroundTime) / n

- cpuUtilization = (totalBurstTime / totalTime) \* 100

**Performance Metrics**

KPIs:

- Average Waiting Time

- Average Turnaround Time

- CPU Utilization

- Throughput

**Benchmark Results (Sample):**

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Avg Wait | Avg Turnaround | CPU Utilization |
| FCFS | 15.2 ms | 28.7 ms | 85% |
| SJF | 8.4 ms | 21.9 ms | 92% |
| RR | 12.1 ms | 25.6 ms | 88% |
| Priority | 10.8 ms | 24.3 ms | 90% |