A

Project Report

On

Project Based Learning (PBL) of Software Engineering

On

CPU SCHEDULER

Submitted in partial fulfillment of the requirement for the VI semester

B.Tech

By

Abhishek Bora (2261065)

Kuldeepak Singh Khati (2261329)

Vansh Pal (2261591)

Vinayak Joshi (2261610)

Under the Guidance

of Mr. Prince Kumar

Assistant Professor

Dept. of CSE

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS
SATTAL ROAD. P.O. BHOWALI

DISTRICT-NAINTAL-263132
2024-25

Graphic Era
Hill University
BHIMTAL CAMPUS

CERTIFICATE

The Report entitled " **CPU SCHEDULER** **",** being submitted by **Abhishek Bora (2261065) s/o Ram Singh, Vansh Pal (2261591) s/o Ravi Pal, Kuldeepak Singh Khati (2261329) s/o Mohan Singh Khati and Vinayak Joshi (2261610) s/o Hem Joshi to Graphic Era Hill University Bhimtal Campus** for the award of Bonafide work carried out by them. They had worked under my guidance and supervision and fulfilled the requirements for the submission of this work report.

(Mr. Prince Kumar)
Faculty-in-Charge

(Dr. Ankur Singh Bist)
HOD, CSE Dept.

Graphic Era Hill University
BHIMTAL CAMPUS

STUDENT’S DECLARATION

We, Abhishek Bora, Vansh Pal, Kuldeepak Singh Khati and Vinayak Joshi, hereby declare the work, which is being presented in the report, entitled “**CPU SCHEDULER”** in fulfillment of the requirement for the award of the degree **Bachelor of Technology (Computer Science)** in the session **2024 - 2025** for semester VI, is an authentic record of our own work carried out under the supervision of **Mr. Prince Kumar** (Graphic Era Hill University, Bhimtal).

The matter embodied in this project has not been submitted by us for the award of any other degree.

Date: ………….

Table of Contents

Chapter 1

Introduction and Problem Statement

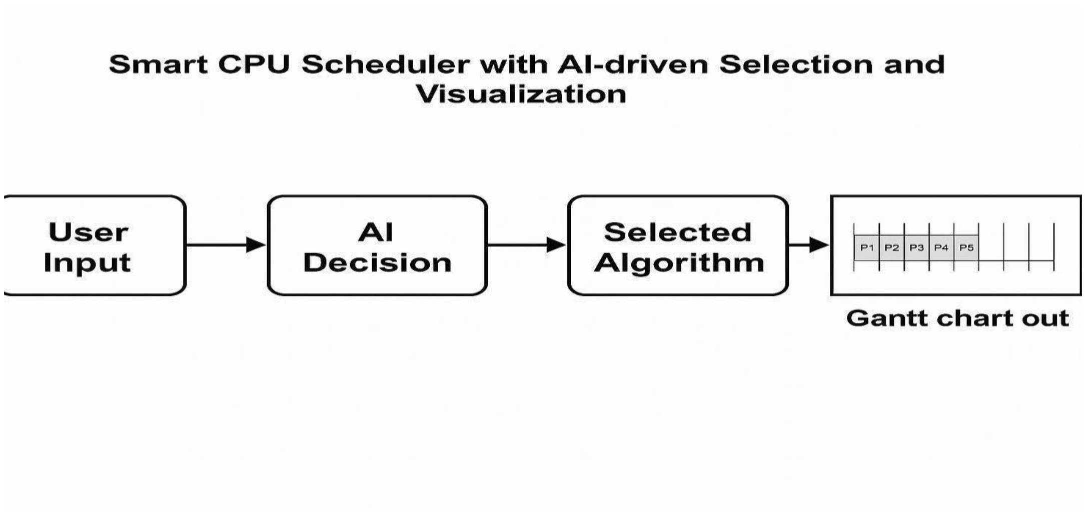
Efficient CPU scheduling plays a vital role in optimizing system performance and resource utilization in modern operating systems. The CPU is the central processing unit responsible for executing multiple processes simultaneously, and an effective scheduling mechanism is essential to ensure fairness, minimize waiting time, reduce turnaround time, and optimize throughput.

As observed by Silberschatz et al. in [1], different CPU scheduling algorithms are suited for different types of workloads. While First Come First Serve (FCFS) is simple and easy to implement, it may result in high waiting times. On the other hand, algorithms like Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Priority Scheduling, and Round Robin (RR) are designed to reduce waiting time and improve responsiveness. However, the effectiveness of each algorithm highly depends on the nature of incoming processes.

In practical systems, deciding which scheduling algorithm to use for a specific scenario is non-trivial. Static selection of a scheduling algorithm may not always yield optimal performance. Therefore, there is a growing need for intelligent systems that can automatically choose the best-suited scheduling approach based on workload characteristics.

To address this challenge, artificial intelligence (AI) can be integrated to build a smart CPU scheduler that can analyze task attributes and decide the optimal scheduling policy dynamically. This system can also provide a real-time visual representation of scheduling using Gantt charts, making it easier for users to understand how processes are being managed by the CPU.

Figure 1.1 illustrates the basic conceptual framework of the proposed smart scheduler with AI-based algorithm selection.



The detailed review of related techniques has been given in [2, 3]. These works highlight the need for dynamic scheduling solutions and the potential of integrating rule-based AI or machine learning techniques to predict the best algorithm based on process metrics like burst time, arrival time, and priority**.**

Problem Statement

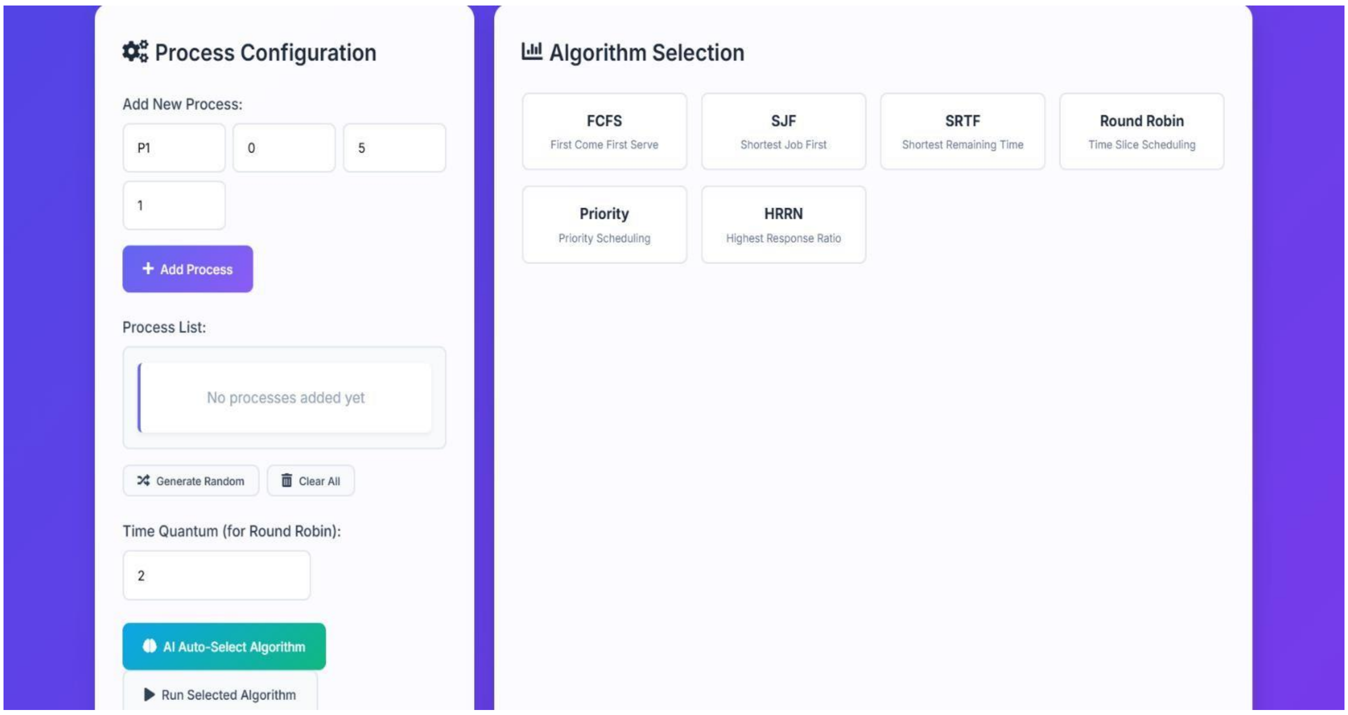
The problem statement for the present work can be stated as follows:

"To design and develop an intelligent CPU task scheduler using Python that analyzes incoming process parameters and automatically selects the most suitable scheduling algorithm using a rule-based AI approach, while also providing real-time Gantt chart-based visualization of the scheduling process."

The key goals of the project are:

- To implement common CPU scheduling algorithms (FCFS, SJF, SRTF, Priority Scheduling, Round Robin, etc.)
- To design a user-friendly GUI for inputting process data and interacting with the system
- To implement a rule-based AI model that selects the appropriate scheduling algorithm based on workload characteristics
- To visualize the scheduling using Gantt charts for clear and real-time process tracking
- To ensure efficient and optimized execution of the scheduling logic

By solving the above problem, the system aims to support educational learning, system design testing, and OS simulation environments where interactive and intelligent CPU scheduling is required.



Chapter 2

Background/ Literature Survey

In the present times, research work is going on in the context of CPU scheduling algorithms, process management, and the integration of artificial intelligence (AI) in operating systems to improve performance, responsiveness, and resource utilization. With the increasing complexity of computing systems, the traditional approach of selecting one fixed scheduling algorithm is no longer optimal. In this chapter, some of the major existing works in these areas have been reviewed.

CPU Scheduling Algorithms

CPU scheduling is a core function of operating systems. Traditional scheduling techniques aim to manage process execution efficiently by maximizing CPU utilization, minimizing waiting time, and ensuring fairness among processes.

- **First-Come First-Serve (FCFS)** is one of the earliest and simplest scheduling methods. However, it suffers from the *convoy effect*, where short processes wait behind long ones.
- **Shortest Job First (SJF)** aims to reduce the average waiting time but may lead to starvation of longer processes.
- **Shortest Remaining Time First (SRTF)**, the preemptive version of SJF, offers better turnaround time but is complex to implement due to the need for continuous process time estimation.
- **Round Robin (RR)** is designed for time-sharing systems. It provides fairness but can increase context switching overhead.
- **Priority Scheduling** allows higher-priority processes to execute first, which can be both preemptive and non-preemptive.
- **Highest Response Ratio Next (HRRN)** is a balanced method that reduces starvation and improves overall responsiveness.
- **Multilevel Queue and Multilevel Feedback Queue Scheduling** are advanced methods that categorize processes into multiple queues with different priorities and allow dynamic movement between queues based on process behavior.

Visualization in Operating Systems

Visualization of CPU scheduling has been introduced in academia and research to help learners understand scheduling behavior. Gantt charts are commonly used to visually represent process execution. Tools like GanttProject and OS simulators provide basic visual interfaces to track process execution timelines. However, most tools lack intelligent features and adaptability.

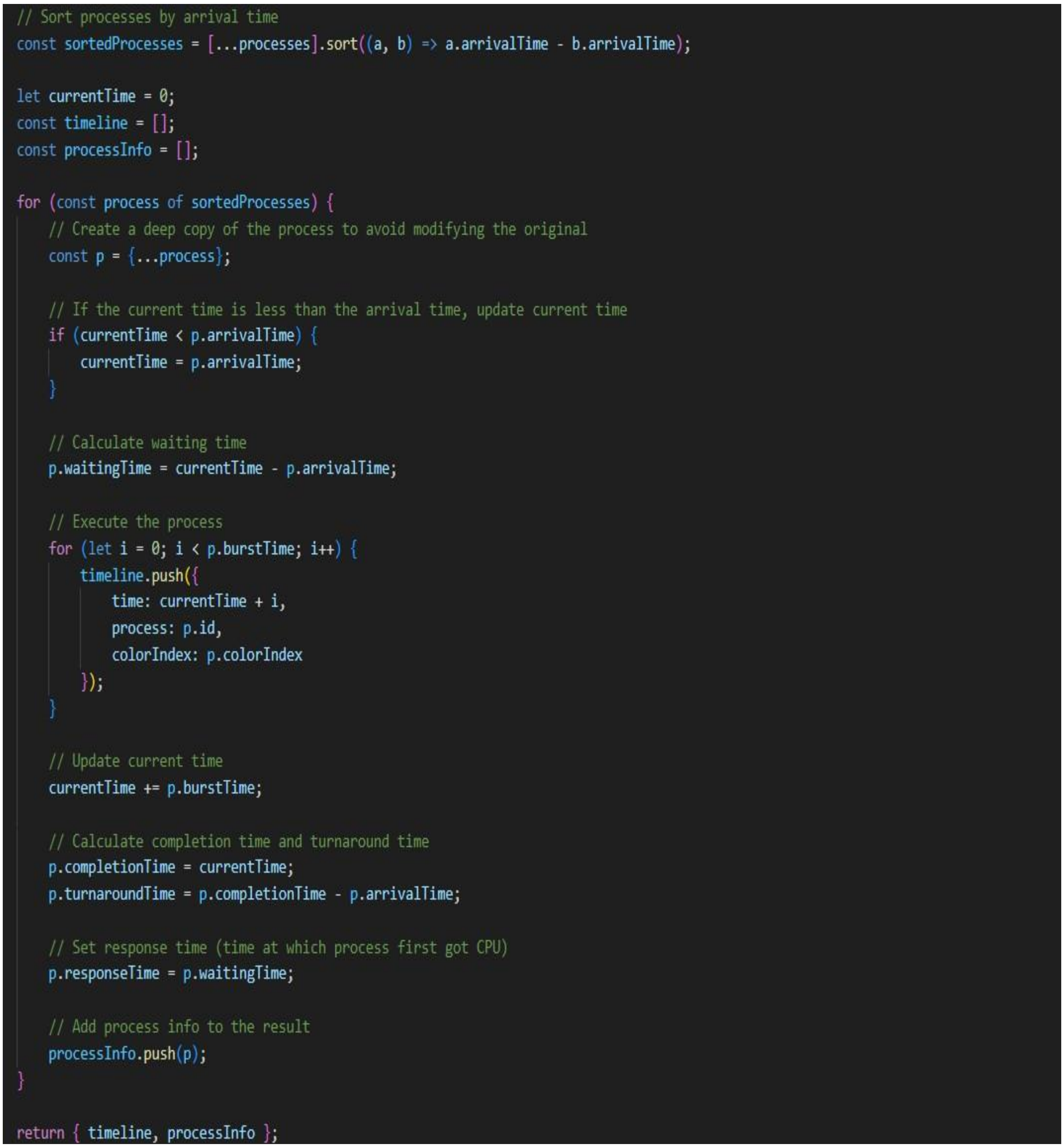
Interactive scheduling simulators like the one developed by Hussain et al. [3] focus on educational use, where users manually select algorithms. These tools help in understanding performance metrics like turnaround time, waiting time, and throughput.

Integration of CPU Scheduling in Educational Tools

Recent developments in operating systems education emphasize the need for practical, interactive tools that help learners grasp complex scheduling concepts. A CPU scheduler project designed for educational use can simulate different scheduling algorithms while providing real-time feedback and performance visualization.

- **Algorithm Simulation**: The scheduler allows users to test and compare classic algorithms like FCFS, SJF, Round Robin, and Priority Scheduling on custom or random process inputs. This provides insight into algorithm-specific characteristics such as waiting time and CPU utilization.
- **Performance Metrics and Visualization**: Integrated tools calculate and display turnaround time, waiting time, and response ratio in a clear, visual format. Gantt chart generation helps learners visualize process execution and context switching.
- **Educational Utility**: In real-time and embedded systems, AI has been used for adaptive scheduling under uncertain workloads. The work by Singh et al. [6] demonstrates adaptive scheduling based on task criticality and deadlines.

Although many visual schedulers exist, few combine interactivity, real-time feedback, and algorithm comparison in a single cohesive platform. Our project aims to address this gap, offering an accessible and educationally rich environment for learning CPU scheduling.



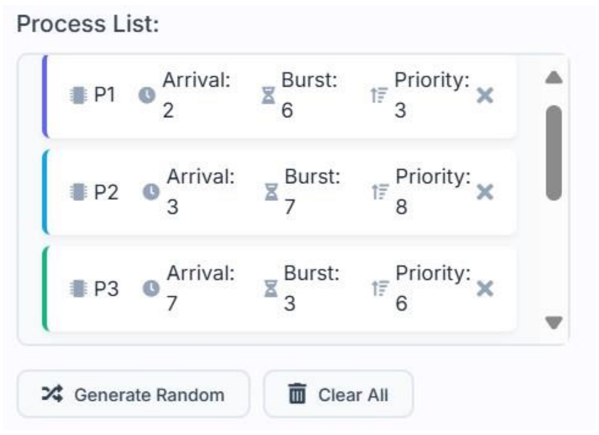
```
// Sort processes by arrival time
const sortedProcesses = [...processes].sort((a, b) => a.arrivalTime - b.arrivalTime);

let currentTime = 0;
const timeline = [];
const processInfo = [];

for (const process of sortedProcesses) {
    // Create a deep copy of the process to avoid modifying the original
    const p = {...process};

    // If the current time is less than the arrival time, update current time
    if (currentTime < p.arrivalTime) {
        currentTime = p.arrivalTime;
    }

    // Calculate waiting time
    p.waitingTime = currentTime - p.arrivalTime;

    // Execute the process
    for (let i = 0; i < p.burstTime; i++) {
        timeline.push({
            time: currentTime + i,
            process: p.id,
            colorIndex: p.colorIndex
        });
    }

    // Update current time
    currentTime += p.burstTime;

    // Calculate completion time and turnaround time
    p.completionTime = currentTime;
    p.turnaroundTime = p.completionTime - p.arrivalTime;

    // Set response time (time at which process first got CPU)
    p.responseTime = p.waitingTime;

    // Add process info to the result
    processInfo.push(p);
}

return { timeline, processInfo };
```

Chapter 3

Objectives

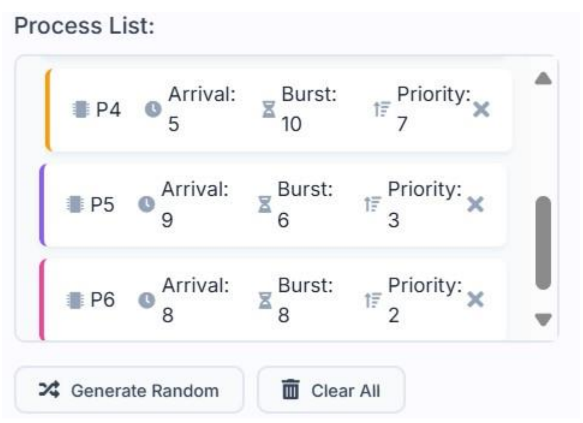
The objectives of the proposed work are as follows:

1. **To implement multiple CPU scheduling algorithms**: Implement commonly used CPU scheduling algorithms including FCFS, SJF, SRTF, Round Robin, Priority Scheduling, HRRN, Multilevel Queue, and Multilevel Feedback Queue using Python.
2. **To develop an interactive GUI-based system:** Design a user-friendly graphical interface for process input, algorithm selection (manual or AI-based), and real-time Gantt chart visualization.
3. **To integrate an AI-based decision system:** Develop a rule-based AI module that analyzes process characteristics and automatically selects the most suitable scheduling algorithm to optimize CPU utilization and minimize waiting time.
4. **To provide real-time visualization and performance metrics:** Display scheduling outcomes using Gantt charts and provide calculated metrics like average waiting time, turnaround time, and throughput for better analysis and learning.
5. **To test and validate system performance across different workloads:** Ensure robustness, accuracy, and responsiveness of the scheduler by testing it on various process loads and validating the correctness of AI-driven decisions.



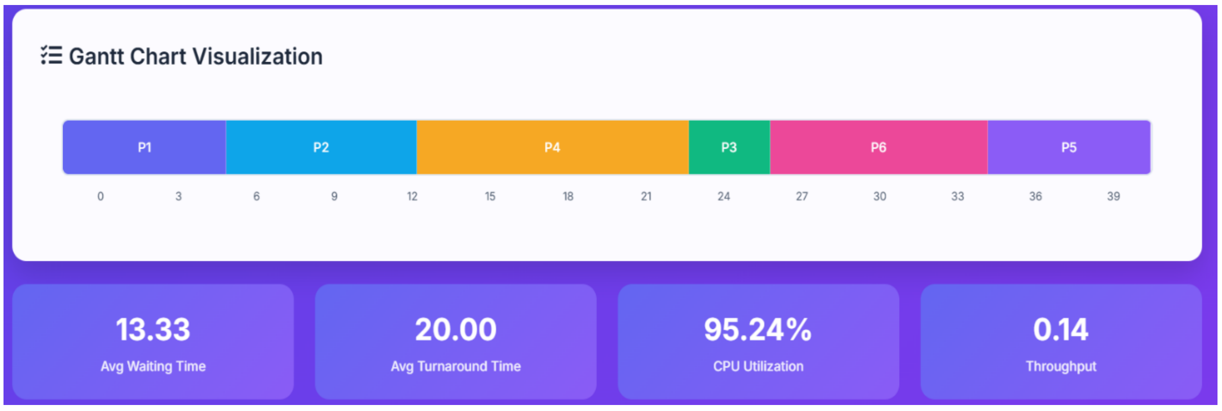
Process List:

	Arrival:	Burst:	Priority:
P1	2	6	3
P2	3	7	8
P3	7	3	6



Process List:

	Arrival:	Burst:	Priority:
P4	5	10	7
P5	9	6	3
P6	8	8	2



Gantt Chart Visualization

13.33 Avg Waiting Time

20.00 Avg Turnaround Time

95.24% CPU Utilization

0.14 Throughput

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements

Sl. No	Name of the Hardware	Specification
1	Processor	Intel Core i5 / AMD Ryzen 5 or higher
2	RAM	Minimum 8 GB
3	Storage	Minimum 256 GB SSD or HDD
4	Display	13” or larger, 1080p recommended
5	Input Devices	Standard Keyboard and Mouse

4.2 Software Requirements

Sl. No	Name of the Software	Specification
1	Operating System	Window 10/11, Linux or MacOs
2	Python	Version 3.8 or above
3	GUI Library (Tkinter/PyQt)	Tkinter(BuiltIn)/PyQt5
4	IDE	VS Code / PyCharm / Jupyter Notebook
5	Visualization Library	Matplotlib / Custom Gantt chart tool
6	AI Rule Engine	Python rule-based logic

Chapter 5

This chapter presents the algorithms considered for CPU scheduling, the AI-based approach for dynamic algorithm selection, and the visualization strategy. The integration of AI with classic scheduling techniques ensures intelligent, adaptive process handling based on input characteristics.

5.1 Traditional Scheduling Algorithms Overview

The system supports implementation of the following scheduling algorithms:

1. **FCFS – First Come, First Serve**
    - Non-preemptive
    - Processes are executed in the order they arrive.
    - Simple but suffers from long average waiting time in many cases.
2. **SJF – Shortest Job First**
    - Non-preemptive
    - Executes the process with the shortest burst time first.
    - Reduces waiting time but causes starvation of longer tasks.
3. **SRTF – Shortest Remaining Time First**
    - Preemptive version of SJF
    - Continuously checks and schedules the process with the least remaining time.
4. **Round Robin (RR)**
    - Preemptive
    - All processes are given a fixed time quantum cyclically.
    - Suitable for time-sharing systems.
5. **Priority Scheduling**
    - Both preemptive and non-preemptive versions exist.
    - Higher-priorityprocesses are scheduled first.
    - Can lead to starvation of low-priority processes.
6. **HRRN – Highest Response Ratio Next**
    - Non-preemptive

- Improves turnaround time and minimizes starvation.

Response Ratio=Waiting Time+Burst TimeBurst Time\text{Response Ratio} = \frac{\text{Waiting Time} + \text{Burst Time}}{\text{Burst Time}}Response Ratio=Burst TimeWaiting Time+Burst Time

7. **Multilevel Queue Scheduling**
   - Non-preemptive
   - Processes are divided into queues based on priority/type and scheduled accordingly.
8. **Multilevel Feedback Queue Scheduling**
   - Preemptive
   - Processes can move between queues based on behavior and execution history.
   - Reduces starvation and adapts to process type dynamically.

5.2 CPU Algorithm Selection Approach

To make the system intelligent, an **AI-driven rule-based module** is integrated to choose the most suitable scheduling algorithm based on process characteristics.

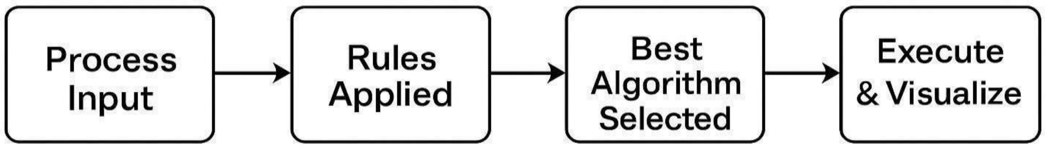


Figure 5.1 Smart Scheduling Decision System

Rule-Based Selection Logic:

Condition	Selected Algorithm
All processes have same arrival time	SJF
Varying priorities with different burst	Priority Scheduling
Small time quantum needed	Round Robin
Processes with very short burst times	SRTF

5.3 Visualization Logic (Gantt Chart Generator)

Once a scheduling algorithm is selected, the output is passed to the visualization module. The Gantt chart plots each process with start and end times for clear visual understanding.

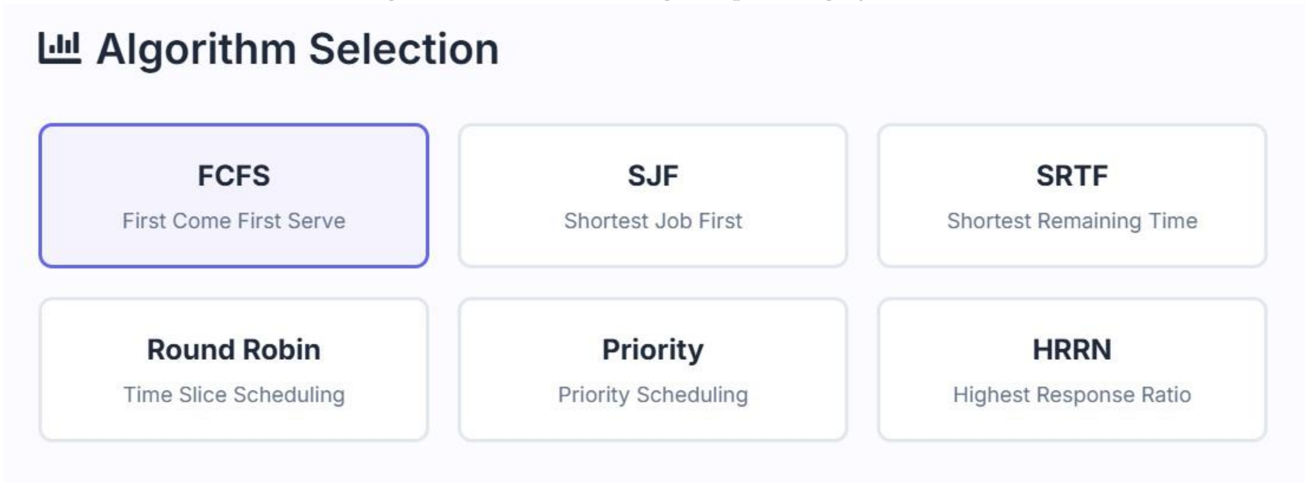
5.4 Evaluation Metrics

To evaluate performance of each algorithm, the following parameters are computed:

- **Average Waiting Time (AWT)**
- **Average Turnaround Time (ATT)**
- **CPU Utilization**
- **Throughput**

5.5 Summary of Approach

This hybrid system blends **traditional scheduling techniques**, **intelligent AI-based selection**, and **interactive visualization** into a single, user-friendly platform. The approach improves educational understanding and decision-making in operating systems simulation tools.



References

[1] S. Sharma and R. Jain, “An Optimized CPU Scheduling Algorithm Based on SJF and Round Robin,” *IEEE International Conference on Computing, Communication & Automation (ICCCA)*, pp. 320–324, April 2016.

[2] D. M. Chaware and M. S. Ali, “Intelligent CPU Scheduling Using Machine Learning Techniques,” *Proc. International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, Jan. 2019, pp. 1052–1056.

[3] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed., Wiley India Pvt. Ltd., 2018.

[4] Python.org – Official Python Documentation. [Online]. Accessed on 1st April 2025: https://docs.python.org/3/

[5] Matplotlib – Visualization with Python. [Online]. Accessed on 1st April 2025: https://matplotlib.org/stable/gallery/index.html

[6] GeeksforGeeks, “CPU Scheduling Algorithms in Operating Systems,” [Online]. Accessed on 30th March 2025: https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/