RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

www.rvce.edu.in
Tel: +91-80-68188100
+91-80-68188111
+91-80-68188112

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CPU Task Scheduler Simulator using Queues

Data Structures and Applications – IS233AI

Experiential Learning (Lab)

REPORT

Submitted by

Avyav Saraf **1RV24IS020**

Amogh S **1RV24IS013**

Under the guidance of

Kavitha S N

Associate Professor, Department of Information Science and Engineering

In partial fulfilment for the award of degree of

Bachelor of Engineering

in

Department of Information Science and Engineering

2025-2026

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,Bengaluru - 560059, Karnataka, India

CERTIFICATE

Certified that the project work titled ***'CPU Task Scheduler Simulator using queues'*** is carried out by **Avyav Saraf (1RV24IS020), Amogh S (1RV24IS013)** who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfilment for the award of degree of **Bachelor of Engineering in Information Science and Engineering** of the **Visvesvaraya Technological University**, Belagavi during the academic year 2025-2026. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of experiential learning work prescribed by the institution for the said degree.

Signature of Guide
Kavitha S N

Signature of Head of the Department
Dr. G S Mamatha

External Viva

Name of Examiners	Signature with Date
1	
2	

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,Bengaluru - 560059, Karnataka, India

DECLARATION

We**, Avyav Saraf & Amogh S,** students of Third semester B.E., department of Information Science Engineering, RV College of Engineering, Bengaluru, hereby declare that Experiential Learning (Lab) titled ***'CPU Task Scheduler Simulator using queues'*** has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Information Science and Engineering** during the academic year 2025-26

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering®, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Name **Signature**

1. Avyav Saraf (1RV24IS020)
2. Amogh S (1RV24IS013)

ABSTRACT

CPU scheduling is a fundamental function of operating systems that influences system performance, responsiveness, and efficient resource utilization. Several scheduling algorithms such as First Come First Served (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin have been widely studied and documented in operating systems literature. However, most existing resources emphasize theoretical analysis with limited interactive visualization, creating a gap between conceptual understanding and practical interpretation. This project aims to bridge this gap by developing an interactive CPU task scheduler simulator that visually demonstrates scheduling algorithms using queue-based data structures and provides real-time performance analysis.

The proposed system is implemented as a web-based simulation tool using Vite and React, ensuring modularity and responsiveness. Queue data structures are employed to manage process execution according to the selected scheduling algorithm. Users can input process parameters such as arrival time and burst time or upload predefined datasets. The simulator dynamically generates Gantt charts and queue animations to illustrate CPU allocation and process flow. Additionally, key performance metrics including CPU utilization, average waiting time, and average turnaround time are computed. The project is entirely software-based and does not involve any hardware components.

The simulator was tested using multiple scheduling scenarios, and the results obtained were consistent with theoretical expectations. The calculated performance metrics closely aligned with standard values reported in operating systems literature, confirming the correctness of the implementation. Interactive visualizations enabled clearer interpretation of execution order, idle periods, and scheduling overhead when compared to traditional static representations.

In conclusion, the CPU Task Scheduler Simulator effectively enhances understanding of CPU scheduling concepts through real-time visualization and performance evaluation. The project demonstrates the practical application of queue data structures and modular design principles. Future enhancements may include support for multi-core scheduling, additional algorithms, and extended analytical capabilities to further improve educational and analytical usefulness.

Chapter 1: Introduction

CPU scheduling is a fundamental responsibility of an operating system, as it determines the order in which processes are allocated processor time. Efficient scheduling is critical for achieving optimal system performance, minimizing waiting time, improving CPU utilization, and ensuring fairness among competing processes [1][2]. Modern operating systems employ various scheduling algorithms, each designed to address specific performance goals and workload characteristics. Understanding the behaviour and trade-offs of these algorithms is essential for students and practitioners studying operating system design.

Despite extensive theoretical coverage of CPU scheduling algorithms, learners often face difficulty visualizing how processes move through ready queues and how scheduling decisions impact execution over time. Traditional teaching methods rely heavily on static examples, tables, and manual Gantt chart construction, which may not effectively convey dynamic scheduling behaviour, creating a gap between theoretical knowledge and practical comprehension [3][4].

The problem addressed in this project is the lack of an interactive and intuitive platform that enables users to simulate CPU scheduling algorithms while simultaneously visualizing process execution and queue behaviour. Without such tools, analysing scheduling performance metrics and understanding real-time process transitions becomes challenging, particularly for complex scenarios involving multiple processes with varying arrival and burst times.

To address this issue, the proposed solution is an interactive CPU Task Scheduler Simulator that models scheduling algorithms using queue-based data structures. The application allows users to input or upload process data, execute selected scheduling algorithms, and observe real-time visualizations in the form of animated Gantt charts and queue transitions. Additionally, the system computes key performance metrics such as CPU utilization, average waiting time, and average turnaround time to support quantitative analysis [5].

The primary objectives of the proposed solution are to enhance conceptual understanding of CPU scheduling algorithms, demonstrate the practical application of queue data structures, and provide an effective educational tool for performance analysis. By integrating simulation, visualization, and metric evaluation within a single platform, the project aims to bridge the gap between theoretical study and practical insight into operating system scheduling behaviour.

Chapter 2: Solution Design

This chapter presents the design aspects of the CPU Task Scheduler Simulator. It describes the overall system architecture, the selection and justification of appropriate data structures, and a high-level flow representation of the system. The focus of this chapter is on explaining what the system is designed to do and how the components interact, rather than detailing execution logic or implementation specifics [1][2].

2.1 System Design and Architecture

The CPU Task Scheduler Simulator is designed as a web-based application following a modular and component-oriented architecture. The system is structured into three major logical components: the user interface module, the scheduling module, and the visualization and analytics module.

The user interface module is responsible for collecting input from the user, including process details such as process identifier, arrival time, burst time, and the selection of the scheduling algorithm. It also provides controls to start, reset, and observe the simulation results.

The scheduling module forms the core of the system. It models CPU scheduling behaviour using abstract data structures and algorithm rules, processing input data to determine the scheduling sequence based on the selected algorithm. The independence of this module from the user interface and visualization logic improves maintainability and scalability [1].

The visualization and analytics module present the output of the scheduling module by converting scheduling results into graphical representations such as Gantt charts and queue visualizations. It also computes and displays performance metrics, enabling users to analyse scheduling behaviour and compare algorithms [3].

The interaction between these modules is designed to be loosely coupled, ensuring clarity in design and ease of future extension.

2.2 Selection and Justification of Data Structures

Queue data structures are the primary data structures used in the proposed system. In CPU scheduling, processes waiting for execution are naturally organized in a ready queue, making queues the most appropriate and efficient choice [1][2].

For algorithms such as First Come First Served (FCFS) and Round Robin, queue operations like enqueue and dequeue directly reflect real-world scheduling behaviour. In the case of Priority Scheduling and Shortest Job First, logical ordering is applied to the queue based on priority values or burst time, while retaining queue-based access principles [4][5].

Basic queue operations such as insertion, deletion, traversal, and front-element access are extensively used. Circular queue behaviour is conceptually employed for Round Robin scheduling to ensure fair time-sharing among processes. Additional data structures such as arrays and objects are used to store process attributes and computed performance metrics.

The use of queue-based data structures ensures efficient scheduling operations and forms the basis for theoretical comparison of scheduling algorithms, simplifies visualization, and closely aligns with standard operating system scheduling models discussed in literature [1][3].

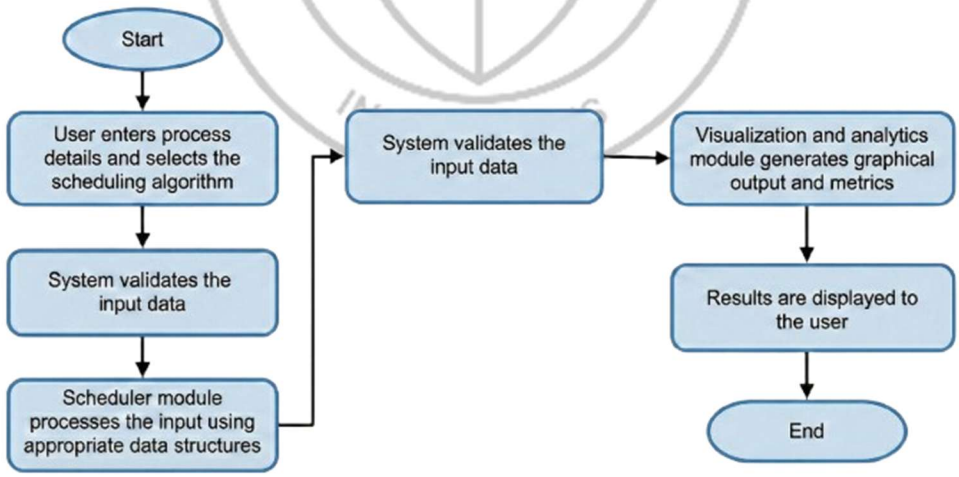

Figure 2.1 Overall workflow of the CPU Task Scheduler Simulator illustrating user input, scheduling logic using queues, and visualization of execution results.

Chapter 3: Implementation Details

This chapter describes the implementation approach adopted for the CPU Task Scheduler Simulator. The implementation focuses on translating the system design into a functional application while maintaining code clarity, modular structure, and ease of maintenance.

The simulator is developed as a client-side web application using Vite and React, enabling fast development and a component-based architecture. The user interface is implemented using reusable components that handle process input, algorithm selection, and simulation control. Input validation is performed to ensure correctness of process parameters before simulation begins.

Scheduling logic is implemented using independent functions that model different CPU scheduling algorithms. Queue-based data structures are used to manage process execution order and scheduling behaviour. Each algorithm processes the validated input data and generates structured outputs such as execution timelines and process completion details. This separation between scheduling logic and user interface improves readability and simplifies future extension of the system.

Visualization and performance analysis are handled through dedicated components that generate Gantt charts, compute scheduling metrics, and provide an analyze comparison feature that presents theoretical time and space complexity, data structures used, and scheduling techniques for each algorithm. State management is organized to track process data and simulation results consistently throughout execution.

Overall, the implementation follows a modular and organized structure, with meaningful naming conventions and minimal code duplication. This approach ensures the simulator remains maintainable, extensible, and suitable for educational and analytical use.

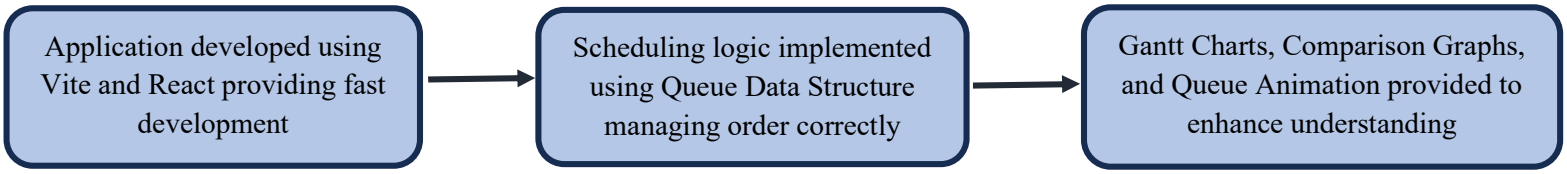

Figure 3.1 Methodology flow of the CPU Task Scheduler Simulator showing interaction between input validation, scheduling algorithms, queue management, and result visualization.

Chapter 4: Results and Discussion

This chapter presents the execution results of the CPU Task Scheduler Simulator and evaluates the correctness and performance of the implemented scheduling algorithms. The discussion focuses on execution outcomes, handling of edge cases, comparative behaviour of algorithms, and potential refinements.

The simulator was tested using multiple process sets with varying arrival and burst times. The execution outcomes were consistent with the theoretical behaviour of the scheduling algorithms. Generated Gantt charts accurately represented process execution order and CPU idle periods. Performance metrics such as average waiting time, average turnaround time, and CPU utilization closely matched values obtained through manual calculation and standard operating systems literature, confirming algorithm correctness [1][2].

Several special and edge cases were evaluated, including identical arrival times, single-process execution, and significant variations in burst time. The simulator handled these cases correctly, maintaining stable execution and accurate metric computation without inconsistencies.

A comparative assessment using identical input datasets and the analyze comparison view highlighted key trade-offs among scheduling algorithms. FCFS showed higher waiting time in certain scenarios, while Round Robin improved fairness at the cost of additional context switching. Shortest Job First achieved lower average turnaround time when burst times were known in advance. These observations align with established theoretical analyses reported in prior studies [3][4].

Based on the analysis, potential improvements include support for advanced scheduling features such as priority aging, real-time scheduling, and multi-core simulation. Similar discussions on scheduling optimizations and practical considerations are reported in review studies and implementation-oriented resources [6][7]. Enhancements in visualization and metric analysis can further improve usability and analytical depth.

The simulator's visualizations clearly illustrated execution order, queue transitions, and CPU idle periods. When combined with performance metrics, these visuals highlighted key algorithmic

differences such as higher waiting time in FCFS and improved fairness in Round Robin. This enabled quick and effective interpretation of scheduling behavior.

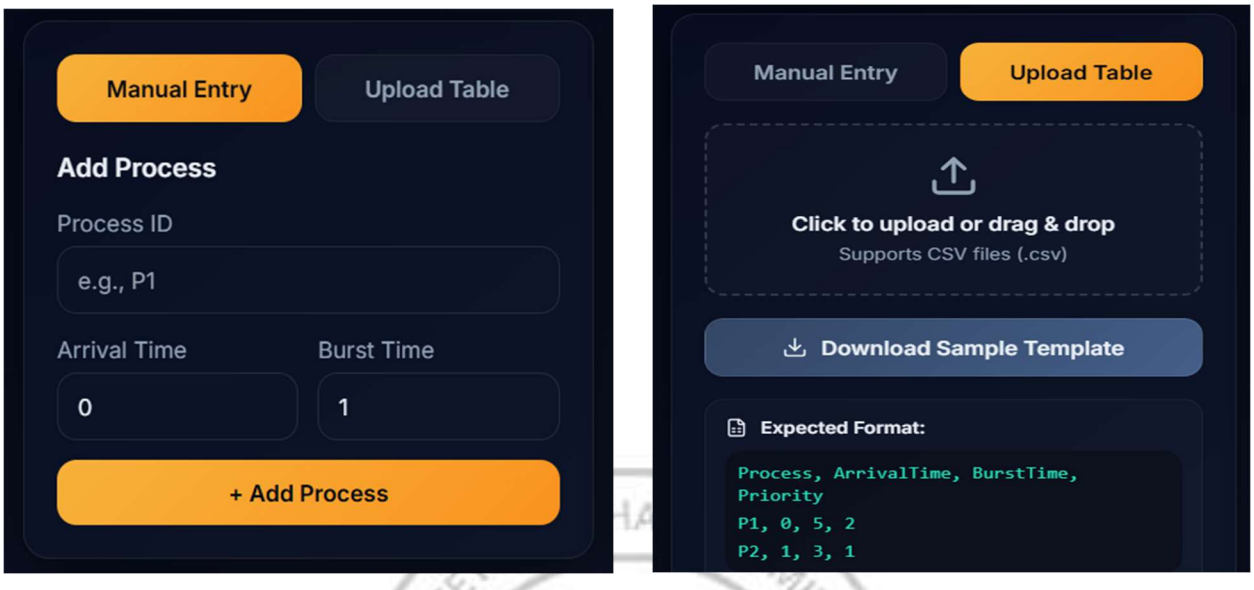

Figure 4.1 User interface demonstrating manual and tabular input options for entering process parameters such as arrival time and burst time.

Figure 4.2: Gantt chart visualization representing the execution sequence of processes generated by the selected CPU scheduling algorithm.

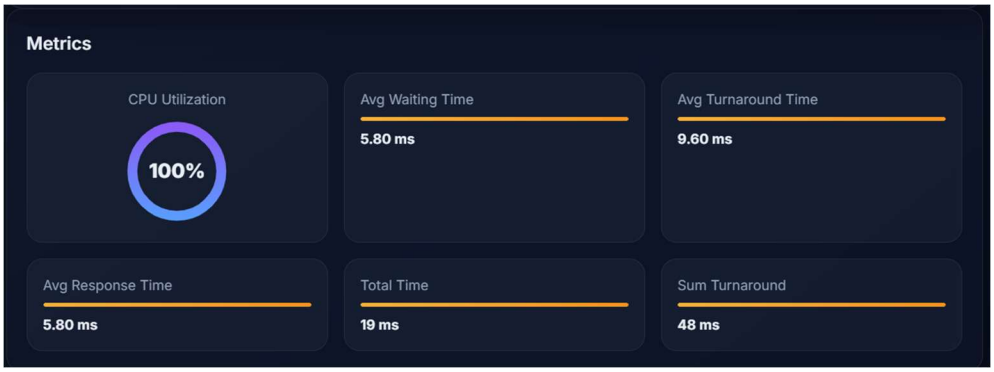

Figure 4.3: Performance metrics analysis displaying average waiting time, average turnaround time, and CPU utilization for the selected scheduling algorithm.

Figure 4.4: Comparative analysis of CPU scheduling algorithms based on theoretical time and space complexity.

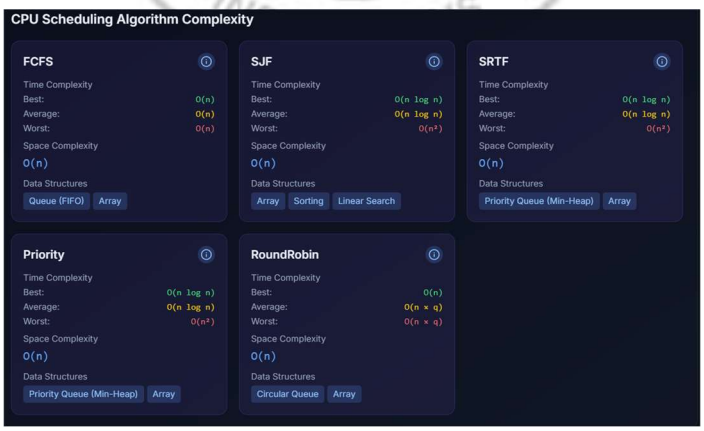

Figure 4.5 Visualization of key data structure parameters, queue ordering and process state information during the scheduling simulation.

Figure 4.6 Detailed performance and behavioural analysis of individual CPU scheduling algorithms under varying process conditions.

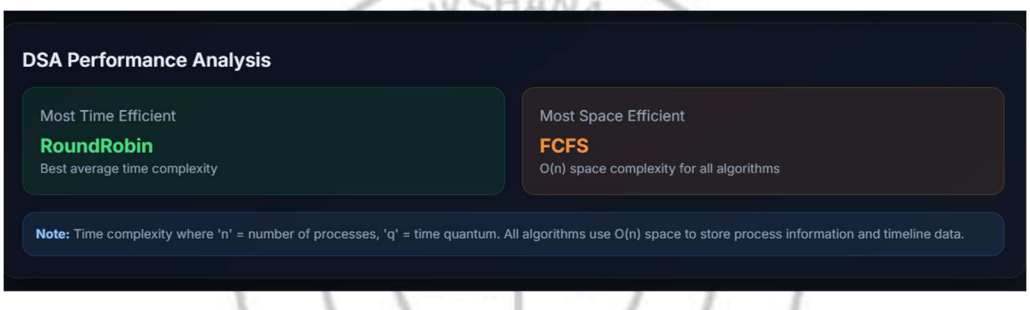

Figure 4.7: Comparison identifying the most suitable scheduling algorithm for a given set of processes based on theoretical computational complexity and design considerations.

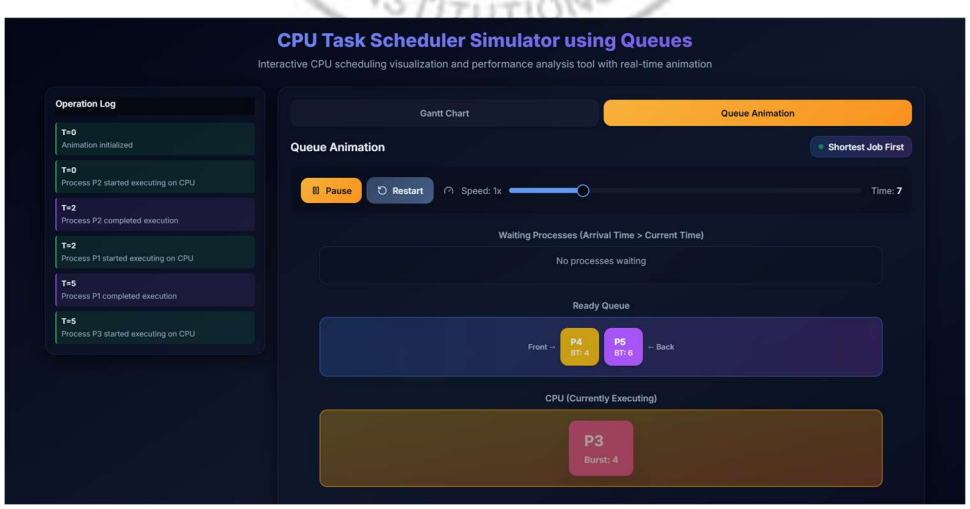

Figure 4.8: Real-time queue animation illustrating process transitions and CPU allocation during scheduling execution.

Chapter 5: Conclusion and Future Scope

This project successfully designed and implemented a CPU Task Scheduler Simulator that demonstrates the practical behaviour of common CPU scheduling algorithms using queue-based data structures. The simulator effectively visualizes process execution through Gantt charts and computes key performance metrics such as average waiting time, average turnaround time, and CPU utilization. The results obtained confirm the correctness of the implementation and highlight the effectiveness of the solution in bridging theoretical concepts with practical understanding.

The proposed solution has strong real-world relevance as CPU scheduling forms the foundation of operating system performance and resource management. The simulator serves as an effective educational and analytical tool for understanding scheduling trade-offs and system behaviour. Its interactive nature enables users to experiment with different workloads and algorithms, making it valuable for academic learning and demonstration purposes.

Future enhancements may include support for advanced scheduling techniques such as priority aging, real-time scheduling, and multi-core CPU simulation. Additional innovations such as improved visualization, statistical analysis, and extensibility for research-oriented experimentation can further increase the applicability of the system in both academic and industry contexts.

References

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Hoboken, NJ, USA: Wiley, 2018.

[2] A. S. Tanenbaum and H. Bos, *Modern Operating Systems*, 4th ed. Boston, MA, USA: Pearson Education, 2015.

[3] N. Goel and R. B. Garg, "A comparative study of CPU scheduling algorithms," *International Journal of Computer Applications*, vol. 45, no. 2, pp. 34–39, 2012.

[4] I. Qureshi, "CPU scheduling algorithms: A survey," *International Journal of Advanced Networking and Applications*, vol. 5, no. 4, pp. 1968–1973, 2014.

[5] H. K. Omar, K. H. Jihad, and S. F. Hussein, "Comparative analysis of the essential CPU scheduling algorithms," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2576–2584, 2021.

[6] N. A. Majedkan, A. J. Ahmed, and L. M. Haji, "CPU scheduling techniques: A review on novel approaches," *Journal of Applied Science and Technology Trends*, vol. 1, no. 1, pp. 10–16, 2020.

[7] GeeksforGeeks, "CPU scheduling in operating systems," https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/, accessed May 2026.