IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 2, February 2026

DOI: 10.17148/IARJSET.2026.13245

JOB SCHEDULING SIMULATOR FOR COMPANIES

Aadith. J. S[1], Dr. R. Praba[2]

Student, B.Sc. Information Technology, Dr. N.G.P. Arts and Science College, Tamil Nadu, India[1]

Associate Professor, B.Sc. Information Technology, Dr. N.G.P. Arts and Science College, Tamil Nadu, India[2]

Abstract: Efficient job scheduling plays a crucial role in improving productivity, reducing operational costs, and optimizing resource utilization in modern companies. Organizations often handle multiple tasks simultaneously across limited resources such as processors, employees, or machines. Improper scheduling may lead to increased waiting time, resource conflicts, reduced throughput, and missed deadlines. This research presents a Job Scheduling Simulator for Companies based on classical and advanced scheduling algorithms. The system simulates real-world task allocation scenarios using algorithms such as First Come First Serve, Shortest Job First, Priority Scheduling, and Round Robin. The simulator evaluates performance metrics including waiting time, turnaround time, throughput, and CPU utilization. The developed application provides a graphical interface for dynamic job input and real-time performance analysis. Experimental results demonstrate that algorithm selection significantly influences system efficiency and resource optimization. The proposed simulator serves as an educational and practical tool for analyzing scheduling strategies in corporate environments.

Keywords: Job Scheduling, CPU Scheduling, Simulation, Resource Allocation, Performance Metrics, Optimization, Operating Systems.

I. INTRODUCTION

In today's competitive and technology-driven corporate environment, organizations are required to manage multiple tasks, operations, and projects simultaneously while utilizing limited resources such as processors, servers, machinery, workforce, or time slots. Efficient job scheduling plays a crucial role in ensuring optimal resource utilization, minimizing delays, improving productivity, and maintaining overall operational stability. Job scheduling refers to the systematic process of allocating available resources to tasks based on predefined criteria or algorithms in order to achieve maximum efficiency and performance. As companies increasingly adopt digital transformation strategies and automated systems, the complexity of workload management has grown significantly. Tasks often arrive dynamically, vary in execution time, possess different priority levels, and require careful coordination to avoid conflicts or bottlenecks. Without effective scheduling mechanisms, organizations may experience excessive waiting times, resource underutilization, task starvation, missed deadlines, and reduced throughput, all of which negatively impact business performance and profitability.

Traditionally, scheduling decisions were made manually by supervisors or managers based on experience and situational judgment; however, manual approaches lack scalability, consistency, and objectivity, especially in large-scale systems where hundreds or thousands of tasks must be processed efficiently. The advancement of computing technologies has led to the development of algorithm-based scheduling methods that automate the allocation process and provide systematic decision-making frameworks. Several classical scheduling algorithms have been widely studied and implemented in computing and operational systems. First Come First Serve (FCFS) executes tasks in the order of arrival and is simple to implement, though it may lead to long waiting times for shorter tasks. Shortest Job First (SJF) improves average waiting time by prioritizing tasks with smaller execution durations, thereby enhancing overall efficiency. Priority Scheduling assigns tasks based on importance levels, ensuring critical operations are completed earlier, while Round Robin (RR) scheduling distributes processing time equally among tasks using fixed time slices, making it suitable for time-sharing environments.

Each scheduling algorithm offers distinct advantages and limitations depending on workload characteristics and organizational objectives, and selecting the most appropriate algorithm for a specific corporate scenario remains a challenging decision. Therefore, simulation-based evaluation becomes essential for analysing algorithm performance before real-world implementation. A Job Scheduling Simulator provides a controlled and analytical platform where various scheduling strategies can be tested using predefined job parameters such as arrival time, burst time, and priority levels. By calculating performance metrics including waiting time, turnaround time, response time, throughput, and resource utilization, the simulator enables comparative analysis and supports data-driven decision-making. The proposed Job Scheduling Simulator for Companies aims to model realistic task allocation scenarios and evaluate multiple scheduling algorithms within a unified framework. Through graphical representations such as Gantt charts and performance comparisons, the system enhances understanding of scheduling behavior and efficiency.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 2, February 2026

DOI: 10.17148/IARJSET.2026.13245

This research contributes to academic learning by providing a practical tool for studying scheduling concepts and also supports corporate environments in optimizing workflow management, improving productivity, and achieving better resource optimization in increasingly complex operational systems.

1.1 PROBLEM STATEMENT

Organizations often face inefficiencies due to improper task allocation and poor scheduling strategies when multiple jobs compete for limited resources. Manual scheduling lacks scalability and may cause delays, increased waiting time, and reduced productivity. Therefore, a systematic simulation-based approach is required to evaluate and compare scheduling algorithms for optimal resource utilization.

1.2 OBJECTIVES

The primary objective of this research is to design and develop a Job Scheduling Simulator capable of analysing and comparing different scheduling algorithms used in corporate environments. The system aims to implement classical algorithms such as FCFS, SJF, Priority Scheduling, and Round Robin within a unified framework. Another objective is to compute and evaluate performance metrics including waiting time, turnaround time, response time, throughput, and resource utilization. The project also seeks to provide graphical visualization through Gantt charts for better understanding. Ultimately, the goal is to create a scalable and user-friendly tool for optimizing task allocation efficiency.

1.3 PROPOSED SYSTEM ARCHITECTURE

The proposed Job Scheduling Simulator for Companies is designed using a modular and layered architecture to ensure scalability, maintainability, and efficient performance analysis. The system architecture consists of five primary modules: Job Input Module, Data Validation and Preprocessing Module, Scheduling Engine Module, Performance Evaluation Module, and Visualization and Reporting Module. Each module performs a specific function while interacting seamlessly with other components to provide a complete scheduling simulation environment. The Job Input Module serves as the entry point of the system, where users provide job-related parameters such as job ID, arrival time, burst time, and priority level. The system supports both manual job entry and predefined datasets to simulate different workload scenarios. Input validation mechanisms are implemented to prevent incorrect or incomplete entries, ensuring accuracy in further processing.

Once the job data is collected, it is passed to the Data Validation and Preprocessing Module, which organizes tasks into structured formats suitable for algorithm execution. This module checks for duplicate entries, validates numerical ranges, sorts jobs based on arrival time when required, and initializes essential parameters such as time quantum in Round Robin scheduling. Proper preprocessing ensures consistency, eliminates ambiguity, and prepares the dataset for accurate simulation. The core component of the architecture is the Scheduling Engine Module, which implements multiple classical scheduling algorithms including First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin (RR). Each algorithm follows its predefined computational logic to determine the order of task execution. The scheduling engine processes jobs dynamically, tracks execution timelines, and generates intermediate outputs such as execution sequences and time calculations.

This modular implementation allows easy extension to include advanced or hybrid scheduling techniques in future enhancements. After job execution is simulated, the Performance Evaluation Module calculates key performance metrics including average waiting time, turnaround time, response time, throughput, and CPU utilization. These metrics provide quantitative measures to evaluate algorithm efficiency and resource optimization. Comparative analysis across algorithms is performed within this module to identify strengths and weaknesses under different workload conditions. The final component, the Visualization and Reporting Module, presents results in an understandable and analytical format. It generates Gantt charts to illustrate execution sequences and graphical representations such as bar charts or tables to compare performance metrics. This visual interpretation enhances clarity and assists users in understanding the impact of algorithm selection. The overall system architecture ensures smooth data flow from job input to result visualization, maintaining separation of concerns across modules. By adopting a modular design approach, the proposed architecture supports scalability, easy maintenance, algorithm expansion, and efficient performance analysis, making it suitable for both academic study and practical corporate scheduling evaluation.

II. METHODOLOGY

2.1 DATASET COLLECTION

The dataset used in this study consists of simulated job records representing real corporate workloads and operational scenarios. Each record includes attributes such as job identification number, arrival time, burst time, priority level, and optional resource requirements. The data may be entered manually by users or generated automatically using predefined test cases to simulate different workload intensities. To ensure realistic analysis, job parameters are designed to reflect practical

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 2, February 2026

DOI: 10.17148/IARJSET.2026.13245

business conditions, including short, medium, and long execution tasks. The dataset is structured in tabular format, where each row represents a job instance and each column corresponds to a specific scheduling attribute. Proper validation is performed to eliminate duplicate entries and incorrect values, ensuring accuracy and consistency. This organized dataset enables systematic evaluation of scheduling algorithms under controlled experimental conditions. Additional scenarios are included to test algorithm performance during high traffic and resource constrained environments for comprehensive comparison analysis purposes only evaluation.

2.2 DATA PREPROCESSING

Data preprocessing is an essential stage in the Job Scheduling Simulator to ensure that input job records are accurate, consistent, and suitable for algorithm execution. Since the dataset may include manually entered or automatically generated job parameters, validation checks are performed to detect missing values, incorrect data types, or negative numerical entries such as invalid arrival or burst times. Duplicate job IDs are identified and removed to prevent redundancy during scheduling. The preprocessing module organizes job data into a structured format and sorts tasks based on arrival time whenever required by specific algorithms. For Round Robin scheduling, the time quantum value is initialized and validated to ensure proper cyclic execution. Priority values are standardized to maintain uniform comparison across tasks. Additionally, the system ensures that all numerical parameters fall within logical operational ranges to avoid computational errors. Proper preprocessing enhances reliability, improves simulation accuracy, and ensures fair performance evaluation across different scheduling algorithms.

2.3 MODEL TRAINING

The model training phase in the Job Scheduling Simulator does not involve machine learning but focuses on implementing and validating classical scheduling algorithms programmatically. In this stage, the preprocessed dataset is supplied to the scheduling engine, where algorithms such as First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin (RR) are executed. Each algorithm follows its predefined logical structure to determine the order of job execution based on parameters like arrival time, burst time, and priority. The system simulates task processing step by step while tracking execution timelines and resource allocation. To ensure correctness, multiple test cases with varying workload conditions are applied. The results are analyzed to verify that each algorithm produces accurate waiting time, turnaround time, and response time calculations. This validation process ensures reliable and consistent scheduling simulation performance across different scenarios.

2.4 JOB SCHEDULING PROCESS AND PERFORMANCE ANALYSIS

Once the scheduling algorithms are successfully implemented and validated, the system proceeds to simulate job execution and analyse performance outcomes. In this phase, the selected scheduling algorithm processes the pre-processed job dataset according to its specific logic. The simulator dynamically tracks execution order, idle time, and task completion sequence. For each job, important time-related metrics such as waiting time, turnaround time, and response time are calculated based on arrival and burst time values. Waiting time represents the duration a job spends in the ready queue before execution, while turnaround time indicates the total time taken from job arrival to completion. Response time measures the time between arrival and the first execution instance, particularly relevant in Round Robin scheduling.

In addition to individual job metrics, overall system performance indicators are computed. Average waiting time and average turnaround time help evaluate algorithm efficiency. Throughput is calculated as the number of jobs completed per unit time, reflecting system productivity. CPU utilization is also measured to determine how effectively processing resources are used during simulation. Comparative analysis is performed across all implemented algorithms to identify which strategy provides optimal performance under specific workload conditions. The results are displayed using structured tables and graphical Gantt charts to clearly visualize execution flow. This analytical approach enables users to understand scheduling behaviour and supports informed decision-making for corporate task management optimization.

III. IMPLEMENTATION DETAILS

The Job Scheduling Simulator is implemented using the Python programming language due to its simplicity, flexibility, and extensive support for algorithm development and visualization. The system is developed in a modular manner to ensure clarity, maintainability, and scalability. Core scheduling algorithms including First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin (RR) are implemented using structured programming logic. Each algorithm operates independently within the scheduling engine, allowing easy comparison and future extension.

The user interface is designed using Tkinter to provide an interactive environment where users can input job parameters such as job ID, arrival time, burst time, and priority. Input validation mechanisms are incorporated to prevent incorrect or incomplete entries. The system processes job data using standard Python data structures such as lists and dictionaries for

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 2, February 2026

DOI: 10.17148/IARJSET.2026.13245

efficient storage and manipulation. Performance metrics including waiting time, turnaround time, response time, throughput, and CPU utilization are computed automatically after execution.

For visualization, Matplotlib is used to generate Gantt charts and comparative graphs, enhancing clarity and analytical understanding. The simulator operates efficiently on standard computing systems without requiring advanced hardware, making it suitable for academic demonstrations and practical scheduling analysis in corporate environments.

3.1 SYSTEM WORKFLOW DESCRIPTION

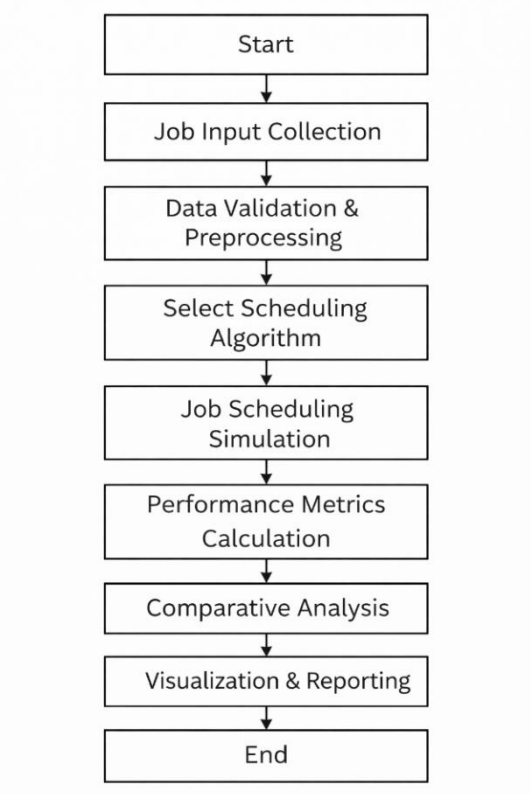
The workflow of the Job Scheduling Simulator begins with the collection of job input data from the user through a graphical interface. Users enter parameters such as job ID, arrival time, burst time, and priority level. Once submitted, the system performs data validation and preprocessing to ensure correctness and remove inconsistencies. After preprocessing, the user selects the desired scheduling algorithm from the available options, including FCFS, SJF, Priority Scheduling, or Round Robin.

The scheduling engine then executes the selected algorithm and determines the order of job allocation. During execution, the system continuously tracks timing parameters required to calculate performance metrics. Upon completion of all jobs, the simulator computes waiting time, turnaround time, response time, throughput, and CPU utilization. Finally, the results are displayed in tabular format along with a Gantt chart for visual representation. This structured workflow ensures accurate simulation and efficient performance evaluation.

3.3 ETHICAL CONSIDERATION

Although the Job Scheduling Simulator primarily handles simulated operational data, ethical considerations remain important in its design and application. The system must ensure fairness in task allocation, particularly when priority-based scheduling is implemented, so that no job experiences indefinite postponement or starvation. Transparency in algorithm logic and performance calculations is essential to maintain trust and accountability. If integrated into real corporate environments, the system should protect sensitive organizational data and maintain confidentiality. Decisions derived from the simulator should support, not replace, human managerial judgment. Proper documentation, unbiased evaluation of algorithms, and responsible usage of scheduling outcomes are necessary to ensure ethical and professional implementation within companies.

3.2 FLOW CHART



IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 2, February 2026

DOI: 10.17148/IARJSET.2026.13245

IV. LIMITATION OF THE SYSTEM

Although the proposed Job Scheduling Simulator provides effective analysis of classical scheduling algorithms, it has certain limitations. The system is restricted to predefined algorithms such as FCFS, SJF, Priority Scheduling, and Round Robin, and does not incorporate advanced optimization techniques or artificial intelligence-based adaptive scheduling methods. The simulator assumes ideal operating conditions and does not fully account for real-world uncertainties such as hardware failures, unexpected job interruptions, or dynamic priority changes during execution. Additionally, the accuracy of performance evaluation depends heavily on the correctness and realism of the input data provided by users. The system does not currently support distributed or cloud-based scheduling environments, which are commonly used in large-scale corporate systems. Scalability may become limited when handling extremely large datasets with thousands of tasks simultaneously. Furthermore, while the simulator provides quantitative metrics, it may not fully capture qualitative organizational factors such as employee workload balance or managerial constraints, which can influence real scheduling decisions.

V. CONCLUSION

The proposed Job Scheduling Simulator for Companies provides a structured and analytical approach to understanding and evaluating task allocation strategies in corporate environments. By implementing classical scheduling algorithms such as First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin (RR), the system demonstrates how different scheduling methods influence overall system performance. Through simulation, the project highlights the impact of algorithm selection on key performance metrics including waiting time, turnaround time, response time, throughput, and CPU utilization.

The modular architecture of the simulator ensures scalability, clarity, and ease of extension, making it suitable for both academic learning and practical organizational analysis. The inclusion of graphical visualization tools such as Gantt charts enhances user understanding of execution flow and scheduling behavior. By providing comparative performance evaluation, the system supports informed decision-making and promotes efficient resource utilization within companies.

However, the effectiveness of the simulator depends on realistic job input data and predefined algorithm constraints. Future enhancements may include integration of advanced optimization techniques, artificial intelligence-based adaptive scheduling, and support for distributed or cloud computing environments. With continuous improvement and responsible implementation, the Job Scheduling Simulator can serve as a valuable tool for improving workflow management, increasing productivity, and supporting data-driven operational strategies in modern organizations.

REFERENCES

[1]. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley.
[2]. Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson.
[3]. Stallings, W. (2018). *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.
[4]. Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1), 46–61. https://doi.org/10.1145/321738.321743
[5]. Coffman, E. G., & Graham, R. L. (1972). Optimal scheduling for two-processor systems. *Acta Informatica*, 1, 200–213.
[6]. Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems* (5th ed.). Springer.
[7]. Baker, K. R., & Trietsch, D. (2009). *Principles of Sequencing and Scheduling*. Wiley.
[8]. Kleinrock, L. (1975). *Queueing Systems, Volume 1: Theory*. Wiley.
[9]. Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. Wiley.
[10]. Buttazzo, G. C. (2011). *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications* (3rd ed.). Springer.
[11]. Rajkumar, R., Lee, I., Sha, L., & Stankovic, J. (2010). Cyber-physical systems: The next computing revolution. *Design Automation Conference*, 731–736.
[12]. Feitelson, D. G. (2015). *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge University Press.
[13]. Leung, J. Y.-T. (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press.
[14]. Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5, 287–326.
[15]. Brucker, P. (2007). *Scheduling Algorithms* (5th ed.). Springer.
[16]. Baruah, S., Mok, A., & Rosier, L. (1990). Preemptively scheduling hard-real-time sporadic tasks on one processor. *Real-Time Systems Symposium*, 182–190.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 2, February 2026

DOI: 10.17148/IARJSET.2026.13245

[17]. Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

[18]. Topcuoglu, H., Hariri, S., & Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3), 260–274.

[19]. Ullman, J. D. (1975). NP-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3), 384–393.

[20]. Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (2001). *Scheduling Computer and Manufacturing Processes* (2nd ed.). Springer.