Software Requirements Specification

for

Process Scheduling Simulator

Version 1.0 approved

Prepared by

Amey Kohale

Rohit Kumar Sharma

Nitesh Kumar Kannaujiya

Himanshu Tambuskar

CDAC Bengaluru

24/12/2023

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to provide detailed software requirements for the development of a Process Scheduling Simulator. This simulator will use React.js and Java Spring Boot for front-end and back-end process. The motive of developing this simulator is to provide a realistic and interactive environment for understanding and analyzing various process scheduling algorithms used in operating systems

1.2 Document Conventions

This document was created on the IEEE template for System Requirement Specification Documen*t*

1.3 Intended Audience and Reading Suggestions

The Simulator is intended for Students and educators. It will help students and educators in gaining in depth knowledge of operating systems and scheduling concepts. By using the simulator as a teaching tool in the courses educators will be able to illustrate concepts of process scheduling with a real time visual representation of the process. They can also customize the simulation to incorporate their course objectives, by demonstrating principles of various scheduling algorithms

1.4 Product Scope

The Simulator will be capable of simulating multiple process scheduling algorithms, such as First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin and Priority Scheduling through a web-based application which will allow the user to create, analyze and visualize scheduling of processes through a responsive user interface

2. Overall Description

2.1 Product Perspective

The Process Scheduling Simulator will have a front end React.js application interacting with a back end through Spring Boot API. The simulator will use REST API calls to transfer data between the front-end and the back-end components.

2.2 Product Functions

- Simulation of various Scheduling Algorithms like First Come First Serve (FCFS), Shortest Job Next (SJN), Round Robin (RR) and Priority Scheduling.
- A user-friendly interface for CRUD operations like creating, editing, and deleting processes.

2.3 User Classes and Characteristics

Students, with limited prior knowledge, utilize the Process Scheduling Simulator as an educational aid, often integrated into coursework for practical insights. Educators, possessing in-depth knowledge, leverage the simulator as a dynamic teaching tool, monitoring student progress. Researchers, equipped with advanced expertise, employ the simulator for experiments and studies, contributing to the field. Developers, adept in programming, use the simulator for testing and debugging scheduling algorithms, ensuring efficient implementations before integration into operating systems.

3. System Features

3.1 Front-end Features

3.1.1 React.js Components

- Interactive user interface components for creating, editing, and deleting processes.
- Visualization components for Gantt chart representation of process execution, understanding and analyzing the results.

3.2 Back-end Features

3.2.1 Java Spring Boot API

- REST API is used for managing processes, simulation control and retrieve data from the front end.
- Logical implementation of the various scheduling algorithms.
- Calculation of performance parameter such as turnaround time, waiting time and throughput during and after simulations.

3.3 Simulation Control

3.3.1 Start, Pause and Stop

Users can start, pause, or terminate the simulation at any point. This allows the users to have a control over the simulation process. These features provide flexibility to the user to tailor the simulation according to their needs.

3.4 Scheduling Algorithms

3.4.1 Implemented Algorithms

The simulator will provide the user with an option to implement major scheduling algorithms such as First Come First Serve (FCFS), Shortest Job Next (SJN), Round Robin (RR), and Priority Scheduling.

3.4.2 Algorithm Selection

Users can select between different scheduling algorithms to compare and understand difference of the wait time, turn-around time, arrival time, burst time, process completion time between two scheduling algorithms during a simulation.

3.5 Visualization

3.5.1 Graphical User Interface (GUI) Design:

The simulator will have an intuitive GUI for users to interact with the visualization. A well-designed interface enhances usability and accessibility.

3.5.2 Gantt Chart

A dynamically updated Gantt chart will be used to display the time line of the processes during the simulation. Gantt charts provide a clear and intuitive way to illustrate when each process is scheduled for execution.

3.5.3 Color-Coding:

Assign distinct colors to different processes or scheduling algorithms. Color-coding enhances clarity and helps users differentiate between processes or algorithmic behaviors.

3.6 System Features

3.6.1 Feature 1 : Home/Start Screen:

- Title: Process Scheduling Simulator
- Input Fields:
    - Number of Processes: [Input Field]
    - Arrival Time: [Input Field]
    - Burst Time: [Input Field]
    - Priority (in case of priority algorithm): [Input Field]
    - Algorithm Selection: [Drop Down]
- Start Simulation Button

3.6.2 Feature 2 : Simulation Visualization:

- Real-time Gantt Chart
- Legend
    - Running Process: [Color]
    - Waiting Process: [Color]
    - Completed Process: [Color]
- Pause/Stop Simulation Button

3.6.3 Feature 3 : Result/Stats Screen

- Performance Parameters
    - Average Turnaround Time: [Value]
    - Average Waiting Time: [Value]
- Reset Simulation Button

4. Interface Requirement

4.1 Front End Interface

The user interface of the application will be developed using React.js. This will ensure a responsive and user-friendly page, which will be compatible with all major web-browsers and different screen sizes. The frontend interface will include components for user-input and data display

4.2 Back-End Interface

The Java Spring Boot back end will expose RESTful APIs for seamless communication with front end. These APIs will handle the different operations such as, processing user requests, managing data and executing logic. Java Spring Boot's versatility and ease of integration males it suitable for developing a reliable and high performance back-end

5. Performance Requirements

Depending upon the scheduling algorithms used different class of process may have different properties The performance of scheduling algorithms used can be compared by factors such as throughput, turnaround, waiting time, and response time.

5.1 Throughput

A measure of the work done by the CPU is the number of processes being executed and completed per unit of time. This is called throughput. The throughput may vary depending on the length or duration of the processes

5.2 Turnaround Time

For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turnaround time is the sum of the periods spent waiting
in the ready queue, executing on the CPU, and doing I/O.

5.3 Waiting Time

A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue.

5.4 Response Time

In an interactive system, turn-around time is not the best criterion. A process may produce some output early and continue computing new results while previous results are being output to the user. Thus, another criterion is the time taken from submission of the process of the request until the first response is produced. This measure is called response time.

6. Other Requirements

Appendix A: Glossary

- Process Scheduling: The technique used by operating systems to manage the execution of processes in a computer system, determining the order in which processes are executed.
- React.js: React is a JavaScript library used for building user interface and single-page applications
- Java Spring Boot: Spring Boot is an open-source Java framework used to create a Micro Service. It provides a faster way to set up and an easier, configure, and run both simple and web-based applications.
- REST API: Representational State Transfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. REST API is a way of accessing web services in a simple and flexible way without having any processing.
- Scheduling Algorithms: Algorithms that dictate the order in which processes are selected to run in a computer system. Common scheduling algorithms include First-Come-First-Serve (FCFS), Round Robin, Shortest Job First (SJF), and Priority Scheduling.

- Gantt Chart: A visual representation of the execution timeline of processes in a system, showing when each process starts and ends.

- Turnaround Time: The total time taken by a process to complete, including both waiting time and execution time.

- Waiting Time: The amount of time a process spends waiting in the ready queue before being executed.

- Round Robin: A preemptive scheduling algorithm where each process is assigned a fixed time slot, and processes are executed in a circular sequence.

- Shortest Job First (SJF): A non-preemptive scheduling algorithm that selects the process with the shortest burst time first.

- Priority Scheduling: A scheduling algorithm where each process is assigned a priority, and the process with the highest priority is selected for execution.

- User Interface (UI): The visual elements and controls through which users interact with the process scheduling simulator.

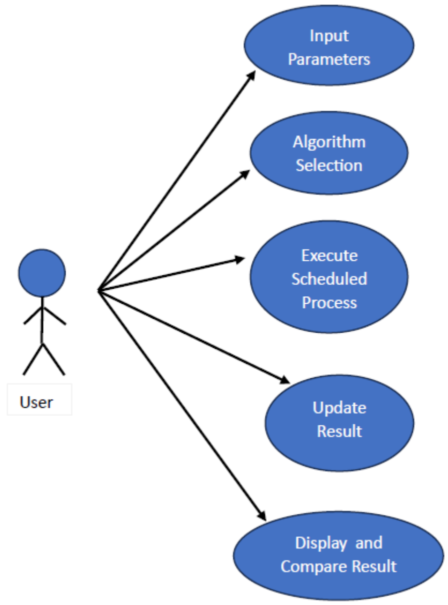
- Scalability: The ability of the simulator to handle a growing number of processes or users without a significant decrease in performance.

- Cross-Platform Support: The capability of the simulator to run on different operating systems without requiring major modifications.

- Open Source: Software that is released with a license that allows anyone to view, use, modify, and distribute the source code.

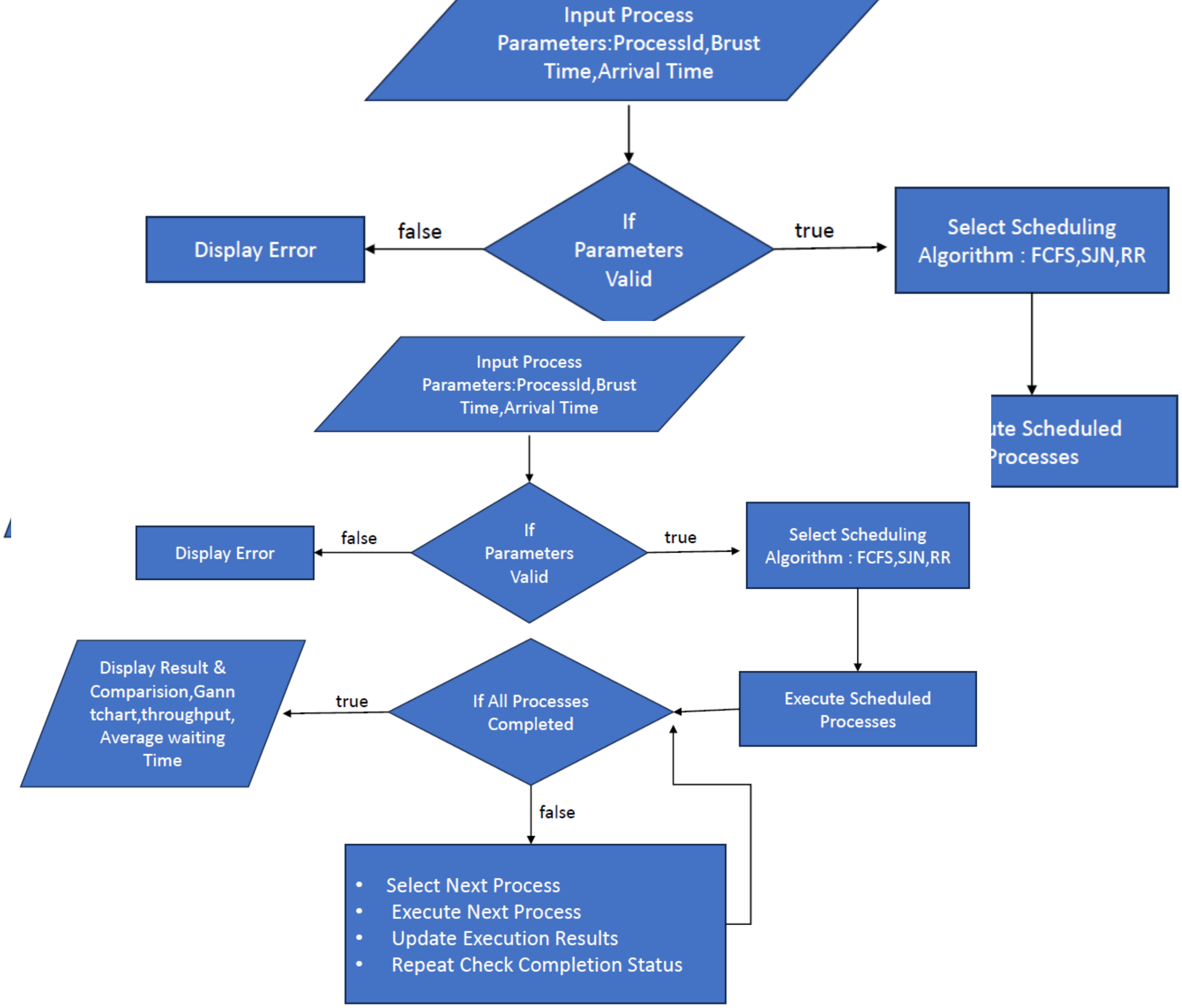
- Real-Time Visualization: The display of simulation results in real-time, providing a dynamic view of the scheduling process as it occurs.

Appendix B: Analysis Models



Use case Diagram

Software


Input Process Parameters:ProcessId,Brust Time,Arrival Time
If Parameters Valid
false
Display Error
true
Select Scheduling Algorithm : FCFS,SJN,RR
ute Scheduled Processes
Input Process Parameters:ProcessId,Brust Time,Arrival Time
If Parameters Valid
false
Display Error
true
Select Scheduling Algorithm : FCFS,SJN,RR
Execute Scheduled Processes
If All Processes Completed
true
Display Result & Comparision,Gann tchart,throughput, Average waiting Time
false
Select Next Process
Execute Next Process
Update Execution Results
Repeat Check Completion Status

Process-Flow Chart