Dharmsinh Desai University

Academic Year: 2022-23
Department: Faculty of Management & Information Science

Subject: Operating System and Linux Programming

Full Name: Akshay Kher.
Roll No: MA027
ID No : 22MAPOS025

Submitted to
Prof. Narayan Joshi
MCA
Department

Student Sign:

Professor Sign

- **Introduction**

The purpose of this app is to provide a CPU scheduling algorithm visualization tool for mobile devices. It allows users to simulate and visualize the behavior of different CPU scheduling algorithms and their effects on the CPU and waiting queue.

★ **The app supports the following algorithms:**

- First-Come-First-Serve (FCFS) Algorithm
- Shortest-Job-First (SJF) Algorithm
- Shortest-Remaining-Time-First (SRTF) Algorithm
- Round Robin Algorithm
- Longest-Job-First (LJF) Algorithm
- Longest-Remaining-Time-First (LRTF) Algorithm
- Priority Scheduling (Preemptive)
- Priority Scheduling (Non-Preemptive)

★ **The app includes the following features:**

- IO Burst support for each algorithm
- Gannt chart visualization of CPU scheduling
- Animation of what happens in CPU and waiting queue
- History Tab to store previously input data

The app's architecture consists of a frontend developed in React Native, which runs on both iOS and Android devices. The frontend communicates with a backend developed in JavaScript that implements the CPU scheduling algorithms and manages data persistence. The app also uses various libraries such as Redux for state management and React Navigation for navigation between screens.

- **User Guide**

★ **Installation:**

a. Download and install the app from the App Store or Google Play Store.
b. Link: https://play.google.com/store/apps/details?id=com.kherakshay007.cpuscheduling
c. Open the app.

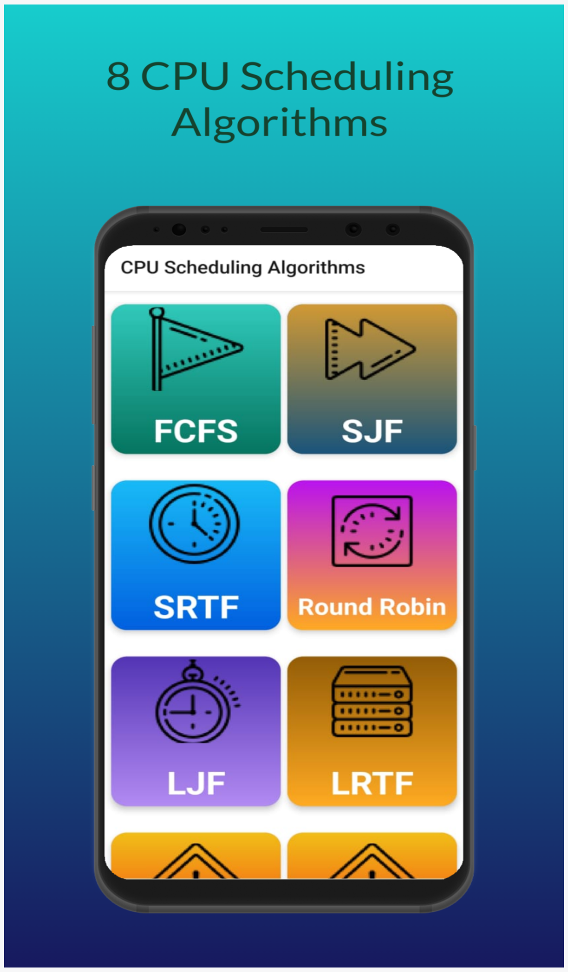
★ **Using the app:**

1. On the home screen, select the CPU scheduling algorithm you want to simulate from the list of supported algorithms.
2. Input the required parameters such as process name, burst time, and priority (if applicable) for each process.
3. Click on the "Run" button to start the simulation.
4. The app will display the Gantt chart visualization of the CPU scheduling, showing the time each process spends in the CPU and waiting queue.
5. The app will also display an animation of what happens in the CPU and waiting queue.
6. Users can click on the "History" tab to access previously input data and simulations.

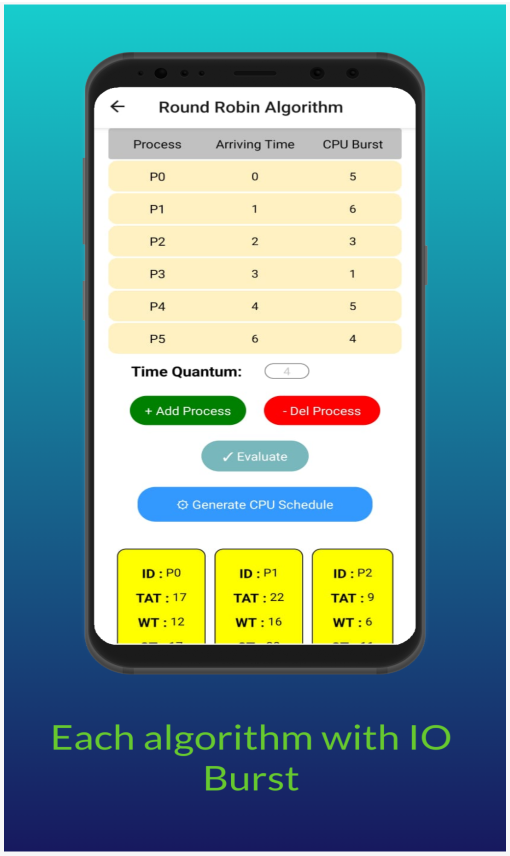
CPU Scheduling React Native Application

★ Screenshots:

(i) Home Screen



(ii) Each Algorithm With Operation:



Round Robin Algorithm

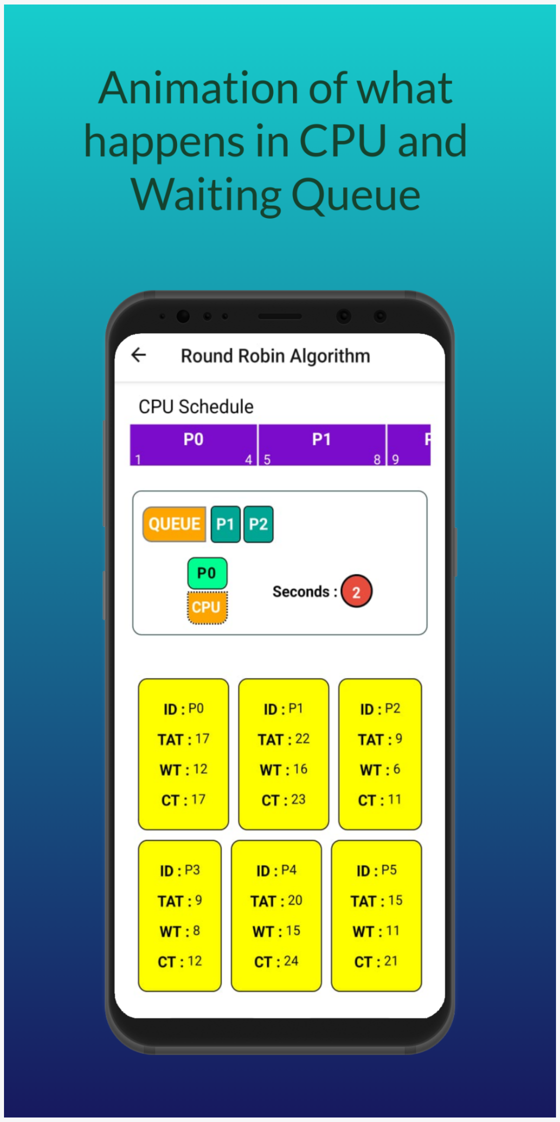
Process	Arriving Time	CPU Burst
P0	0	5
P1	1	6
P2	2	3
P3	3	1
P4	4	5
P5	6	4

Time Quantum: 4

ID : P0	ID : P1	ID : P2
TAT : 17	TAT : 22	TAT : 9
WT : 12	WT : 16	WT : 6

CPU Scheduling React Native Application

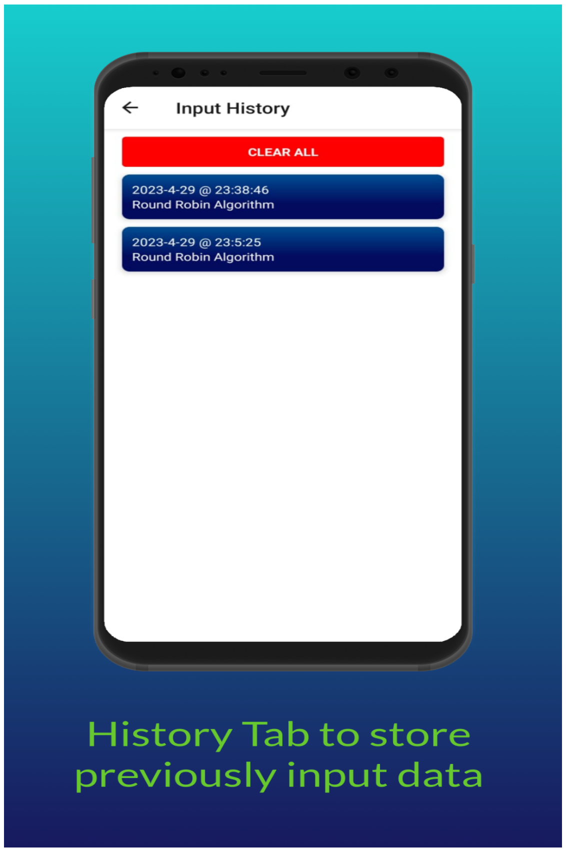
(iii) Animation of What Happens in CPU and Waiting Queue



ID : P0	ID : P1	ID : P2
TAT : 17	TAT : 22	TAT : 9
WT : 12	WT : 16	WT : 6
CT : 17	CT : 23	CT : 11
ID : P3	ID : P4	ID : P5
TAT : 9	TAT : 20	TAT : 15
WT : 8	WT : 15	WT : 11
CT : 12	CT : 24	CT : 21

CPU Scheduling React Native Application

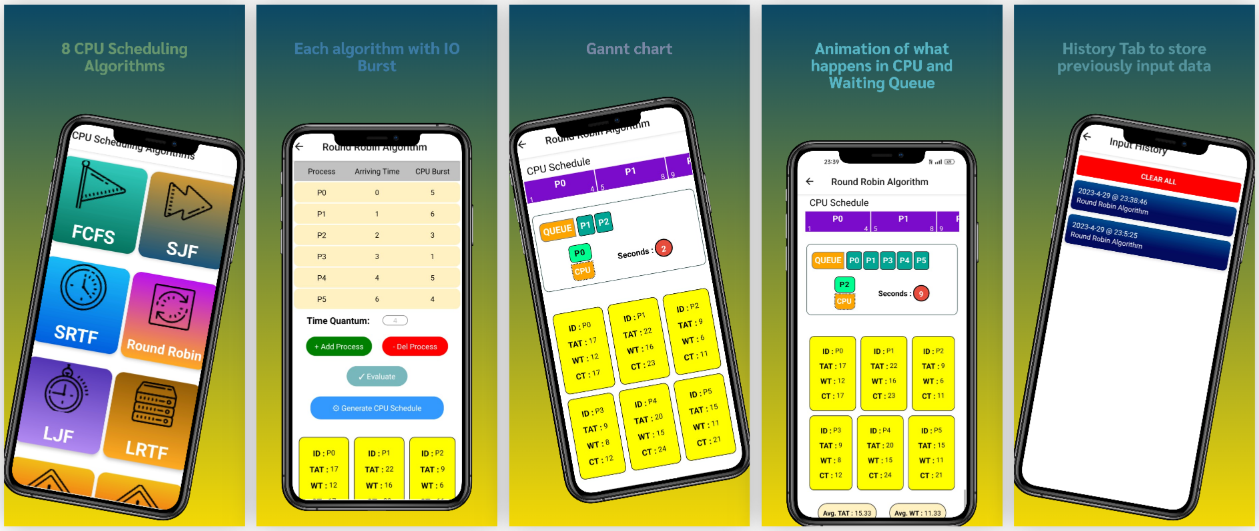
(iv) Input History



2023-4-29 @ 23:38:46
Round Robin Algorithm

2023-4-29 @ 23:5:25
Round Robin Algorithm

(V) One ScreenShot



Process	Arriving Time	CPU Burst
P0	0	5
P1	1	6
P2	2	3
P3	3	1
P4	4	5
P5	6	4

ID	TAT	WT	CT
P0	17	12	17
P1	22	16	23
P2	9	6	11
P3	9	8	12
P4	20	15	24
P5	15	11	21

Avg. TAT : 15.33 Avg. WT : 11.33

2023-4-29 @ 23:38:46
Round Robin Algorithm

2023-4-29 @ 23:5:25
Round Robin Algorithm

- <u>**CPU Scheduling Algorithms**</u>

  - ★ **First-Come-First-Serve (FCFS) Algorithm:**
    - This algorithm schedules processes in the order they arrive. It is non-preemptive and has a simple implementation. However, it may result in long waiting times for processes with long burst times. A visualization of this algorithm shows that the CPU runs each process to completion before starting the next process.

  - ★ **Shortest-Job-First (SJF) Algorithm:**
    - This algorithm schedules processes with the shortest burst time first. It is non-preemptive and aims to minimize the average waiting time of processes. However, it requires knowledge of the burst times of all processes in advance, which may not be practical in some situations. A visualization of this algorithm shows that the CPU runs the shortest job first, even if it arrives after other processes.

  - ★ **Shortest-Remaining-Time-First (SRTF) Algorithm:**
    - This algorithm is similar to SJF but is preemptive, meaning that it can interrupt a running process if a new process with a shorter burst time arrives. It aims to further reduce the average waiting time of processes. However, it may result in a large number of context switches and high overhead. A visualization of this algorithm shows that the CPU runs the process with the shortest remaining time, and may switch to another process if a shorter one arrives.

  - ★ **Round Robin Algorithm:**
    - This algorithm allocates a fixed time slice, known as a time quantum, to each process in a round-robin fashion. It is preemptive and aims to provide fair sharing of the CPU among processes. However, it may result in high overhead due to frequent context switching, and long processes may still have to wait for their turn to use the CPU. A visualization of this algorithm shows that each process is allocated a time quantum and the CPU switches

to the next process after the time quantum expires.

★ **Longest-Job-First (LJF) Algorithm:**
  ○ This algorithm schedules processes with the longest burst time first. It is non-preemptive and aims to minimize the average turnaround time of processes. However, it may result in long waiting times for processes with short burst times, and it requires knowledge of the burst times of all processes in advance. A visualization of this algorithm shows that the CPU runs the longest job first, even if it arrives after other processes.

★ **Longest-Remaining-Time-First (LRTF) Algorithm:**
  ○ This algorithm is similar to LJF but is preemptive, meaning that it can interrupt a running process if a new process with a longer burst time arrives. It aims to further reduce the average turnaround time of processes. However, it may result in a large number of context switches and high overhead. A visualization of this algorithm shows that the CPU runs the process with the longest remaining time, and may switch to another process if a longer one arrives.

★ **Priority Scheduling (Preemptive):**
  ○ This algorithm schedules processes based on their priority level. It is preemptive and allows higher priority processes to interrupt lower priority processes. It aims to give priority to important processes. However, it may result in lower priority processes waiting for a long time, and it requires a good understanding of the relative importance of processes. A visualization of this algorithm shows that the CPU runs higher priority processes first, and may switch to lower priority processes if a higher one is blocked.

★ **Priority Scheduling (Non-Preemptive):**
  ○ This algorithm is similar to the preemptive version but is non-preemptive, meaning that a running process cannot be interrupted. It aims to give priority to important processes but may result in long waiting times for lower priority processes. A visualization of this algorithm shows that the CPU runs the highest priority process first and waits for it to complete before running the next one.

- **Features**
  - ★ **IO Burst Feature:**
    - The IO burst feature is an additional burst time that simulates input/output (IO) operations that a process may require during execution. The IO burst time is a period when a process is blocked and cannot use the CPU. The addition of IO burst times affects CPU scheduling by introducing a wait time that a process must go through before it can use the CPU again. The IO burst time is an important consideration in CPU scheduling algorithms as it can significantly affect the overall turnaround time of processes.
  - ★ **Gannt Chart Feature:**
    - The Gannt chart is a graphical representation of the CPU scheduling algorithm that displays the timeline of processes' execution. The chart shows when each process is running, blocked, or waiting. It helps users visualize the order and duration of processes' execution, which is helpful in analyzing the performance of the CPU scheduling algorithm. The Gannt chart also helps identify which processes are causing delays and how long the delays are.
  - ★ **Animation Feature:**
    - The animation feature shows what happens in the CPU and waiting queue during the execution of CPU scheduling algorithms. It provides a visual representation of how processes move through the CPU and waiting queue, and how they are affected by the scheduling algorithm. The animation feature is helpful in understanding how the CPU scheduling algorithm works and how it affects processes' execution.
  - ★ **History Tab Feature:**
    - The history tab feature allows users to store previously input data, making it easier to re-run previous simulations. It helps users compare the performance of different scheduling algorithms or configurations. The history tab feature can also be used to analyze the performance of a particular scheduling algorithm under different scenarios. Users can store multiple sets of input data in the history tab and switch between them as needed. The history tab feature makes it easier to reproduce and analyze previous simulations.

CPU Scheduling React Native Application

- **Technical Details**
  - ★ **Programming languages and libraries:**
    - The app is developed using React Native, a popular framework for building native mobile apps using JavaScript. The app also uses various libraries and tools, including Redux for state management, D3.js for data visualization, and Moment.js for time formatting.
  - ★ **Code structure:**
    - The app's code structure is organized into different components that handle specific tasks, such as the CPU scheduling algorithms, IO burst, Gannt chart, animation, and history tab. The app also uses Redux for state management, which allows for easy access and manipulation of app state data.
  - ★ **CPU scheduling algorithm implementation:**
    - Each CPU scheduling algorithm is implemented as a separate module that takes in the input data and returns the order of execution for each process. The app uses a switch statement to select the appropriate algorithm based on the user's selection. The algorithms are designed to work with both normal burst time and IO burst time.
  - ★ **Challenges and limitations:**
    - One of the biggest challenges during the development process was optimizing the performance of the app. CPU scheduling algorithms can be computationally intensive, especially when dealing with large sets of input data. The app was optimized using various techniques, including reducing unnecessary computations and using memoization. Another challenge was implementing the animation feature, which required precise timing and synchronization with the CPU scheduling algorithm. The app also has some limitations, such as only supporting a limited set of CPU scheduling algorithms and input data. Future versions of the app may include additional algorithms and features.

- ## Future Developments

We are always looking for ways to improve the app and add new features. Some possible future developments include:

1. Adding more CPU scheduling algorithms: We are looking into adding more scheduling algorithms, such as multi-level feedback queues, lottery scheduling, and earliest deadline first.
2. Enhancing the animation feature: We plan to improve the animation feature by adding more visual elements and allowing users to customize the animation speed and colors.
3. Adding support for different platforms: Currently, the app is only available for iOS and Android devices. In the future, we plan to expand support to other platforms, such as web browsers and desktop operating systems.

We welcome feedback and suggestions from users on how to improve the app. If you have any ideas or feature requests, please let us know. We are committed to making the app as useful and user-friendly as possible.

- ## Conclusion

In conclusion, this documentation provides a comprehensive guide to using the CPU scheduling algorithm visualization app. We covered the purpose and goals of the app, the supported algorithms and features, a user guide, technical details, and future developments.

We hope this documentation has been helpful in understanding how to use the app and how it works. We encourage users to provide feedback and suggestions for improvements so that we can continue to enhance the app and make it more useful for everyone.

Thank you for using the CPU scheduling algorithm visualization app. If you have any questions or need assistance, please contact us .

Github Project Repo Link : https://github.com/akshay0077/CPU-Scheduling-App

Play Store Link : https://play.google.com/store/apps/details?id=com.kherakshay007.cpuscheduling

Linkedin Link: https://www.linkedin.com/in/kherakshay/