International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Operating System Demo Kit

Shreya Shetake[1], Vanshhita Kachare[2], Aditi Kadam[3], Snehal Mohite[4], Er. Yogita Khandagale[5].

[1,2,3,4] Student, Information Technology, Vidyalankar Polytechnic, Wadala
[5]Faculty, Information Technology, Vidyalankar Polytechnic, Wadala

ABSTRACT:

In this project, we have developed a comprehensive web application designed as a demo kit for understanding and testing various operating system concepts. The application provides an interactive platform for students, educators, and researchers to experiment with and visualize key operating system algorithms. It includes modules for scheduling algorithms, Linux kernel terminator simulation, page replacement algorithms, and deadlock handling mechanisms. By offering real-time testing and analysis, the tool enhances learning by allowing users to see the behaviour of different algorithms under different conditions. This platform is particularly beneficial for universities and educational institutions, enabling users to better understand complex operating system concepts and their practical implementations. The user-friendly interface and detailed outputs make it an invaluable resource for students studying operating systems.

Keywords: Operating System Simulation , CPU Scheduling Algorithms, Page Replacement Strategies, Deadlock Handling, Real-time Visualization, Educational Tool.

Introduction:

This web application is an educational tool designed to help users understand and test key operating system concepts. It features interactive modules for simulating CPU scheduling algorithms, Linux kernel process termination, page replacement algorithms, and deadlock handling mechanisms. The platform allows students and educators to visualize and experiment with these algorithms in real-time, enhancing the learning experience by bridging theoretical knowledge with practical application. It serves as an invaluable resource for universities, making complex operating system topics easier to grasp. Traditional methods lack practical, real-time simulations for understanding complex OS algorithms like scheduling, page replacement, and deadlock handling.

Review of Literature:

We conducted a study on an application called 'CPU Scheduling,' available on the Play Store, and identified the following drawbacks. Here's how our project addresses these issues:

CPU Scheduling

1. **Limited Visualizations for Complex Concepts**

The "CPU Scheduling" app visualizes CPU scheduling using basic bar charts (Gantt charts) but may not provide in-depth visualizations for more complex concepts such as deadlock handling or page replacement algorithms. It also may not fully explain how the algorithms interact with real-world scenarios.

How Our Project Covers It:

- Providing interactive Gantt charts for CPU scheduling that allow users to tweak process durations and priorities.
- Visualization of page replacement algorithms like LRU, FIFO, and Optimal with a real-time memory management simulation.
- Detailed visual representations for deadlock detection, avoidance, and recovery, allowing users to simulate real-world OS problems and solutions.

2. **Lack of Comprehensive Educational Content**

The app seems to focus primarily on providing algorithms for CPU scheduling without diving deeply into the theory or detailed explanations behind these algorithms. The user may not fully understand why a certain algorithm is preferred in a particular scenario.

How Our Project Covers It:

- In-depth explanations for each scheduling algorithm, page replacement strategy, and deadlock handling technique. These explanations can cover the underlying theory, advantages, and disadvantages of each method.
- Textual and graphical explanations to show how each algorithm behaves in different conditions and under various workloads. This can be achieved by including brief lessons or tutorials before or after the user tests each algorithm.

3. **No Deadlock Handling and Page Replacement Algorithms**

While the app provides CPU scheduling algorithms, it does not seem to cover page replacement algorithms or deadlock handling,which are crucial OS concepts. These topics are important for anyone learning about operating systems, as they deal with resource management and process synchronization.

How Our Project Covers It:

- Page replacement algorithms like FIFO, LRU, Optimal, and Clock algorithms, providing simulations for how these algorithms handle memory management.
- Deadlock detection, prevention, and recovery strategies with corresponding visualizations to help users understand how deadlocks occur and how they can be resolved in an OS.
- Interactive exercises that allow users to try to create deadlock situations or resolve them using different methods (e.g., Banker's Algorithm).

Methodology:

The OSY Demo Kit was developed using a combination of technologies:

- **Frontend**: React.js
- **Backend**: Python

Key Features:

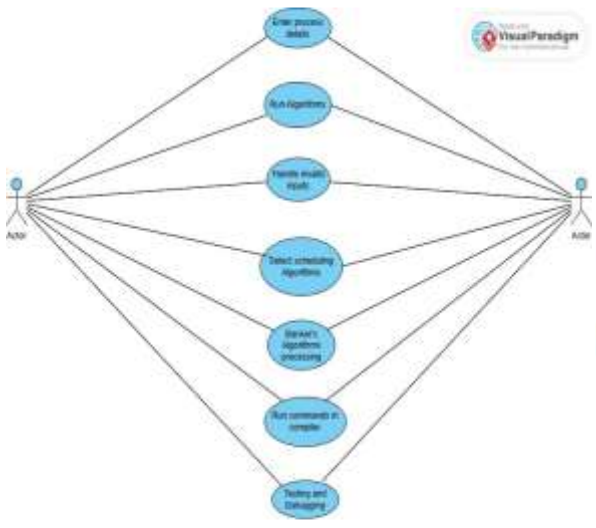
Page Replacement – Simulates FIFO , LRU, Optimal, and Clock algorithms with real-time memory visualization.

CPU Scheduling – Interactive Gantt charts for FCFS, SJF, Priority, and Round Robin with performance analysis.

Linux Terminal Simulation – Executes basic Linux shell commands with process management features.

Deadlock Handling – Detects, prevents, and resolves deadlocks with interactive exercises and Banker's Algorithm.

Diagrams:



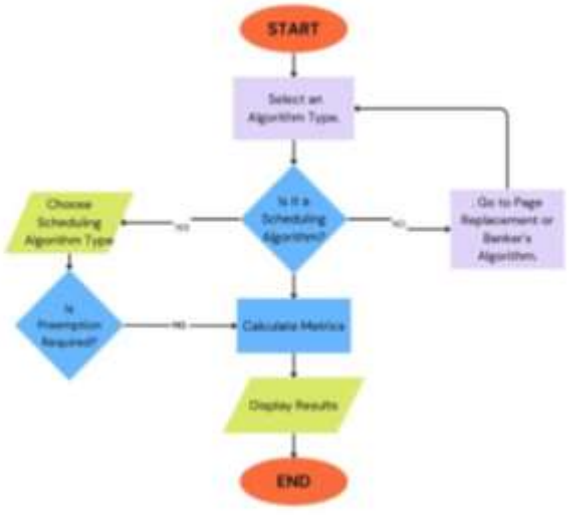
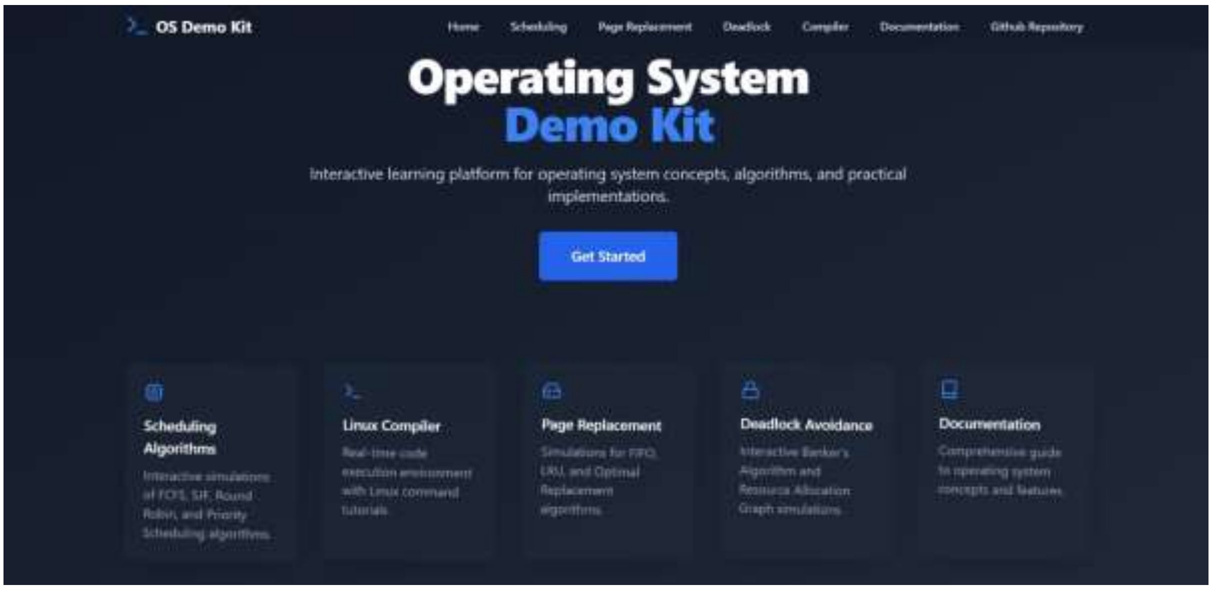

START
Select an Algorithm Type.
Is it a Scheduling Algorithm?
YES
NO
Choose Scheduling Algorithm Type
Go to Page Replacement or Banker's Algorithm.
Is Preemption Required?
NO
Calculate Metrics
Display Results
END

Fig:1 Use Case Diagram Fig:1.1 Flowchart

Results:


OS Demo Kit

Home Scheduling Page Replacement Deadlock Compiler Documentation Github Repository

Operating System Demo Kit

Interactive learning platform for operating system concepts, algorithms, and practical implementations.

Get Started

Scheduling Algorithms
Interactive simulations of FCFS, SJF, Round Robin, and Priority Scheduling algorithms.

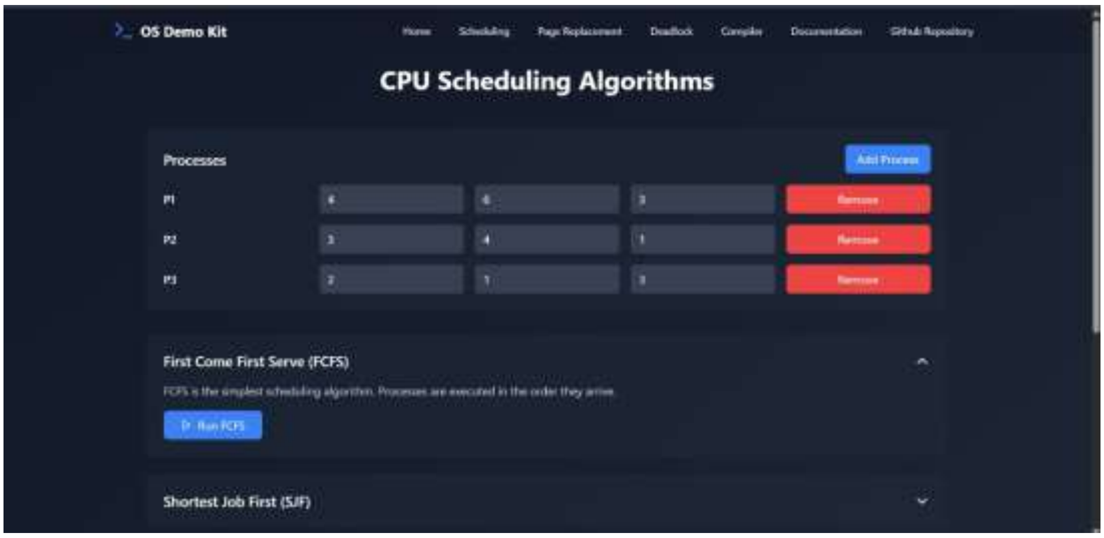
Linux Compiler
Real-time code execution environment with Linux command tutorials.

Page Replacement
Simulations for FIFO, LRU, and Optimal Replacement algorithms.

Deadlock Avoidance
Interactive Banker's Algorithm and Resource Allocation Graph simulations.

Documentation
Comprehensive guide to operating system concepts and features.

Fig.1. Home Page



CPU Scheduling Algorithms

Processes

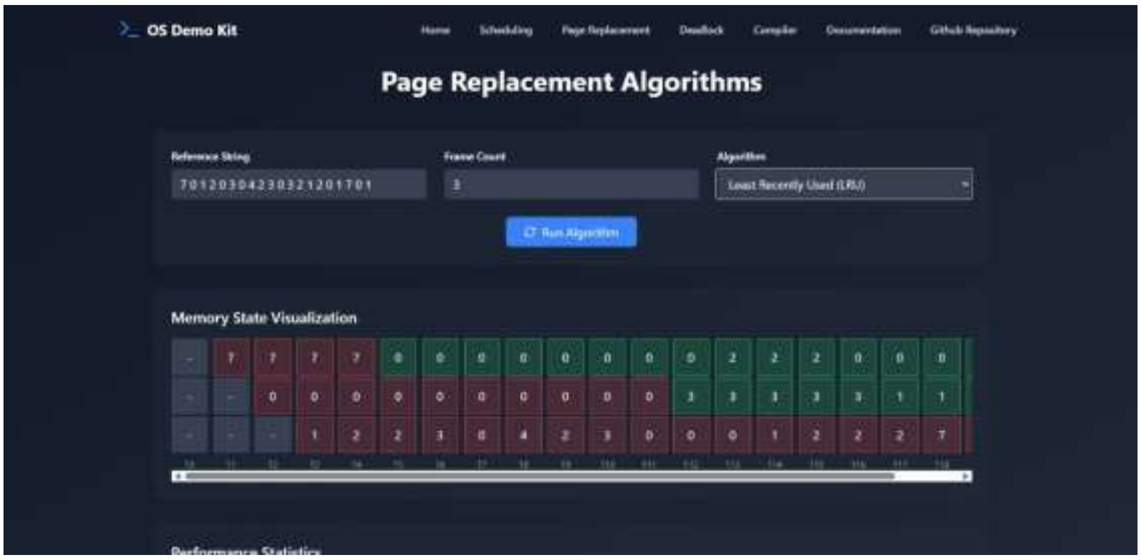
P1	4	6	3
P2	3	4	1
P3	2	1	3

First Come First Serve (FCFS)

FCFS is the simplest scheduling algorithm. Processes are executed in the order they arrive.

Shortest Job First (SJF)

Fig.2. CPU Scheduling Algorithm



Page Replacement Algorithms

Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Frame Count: 3

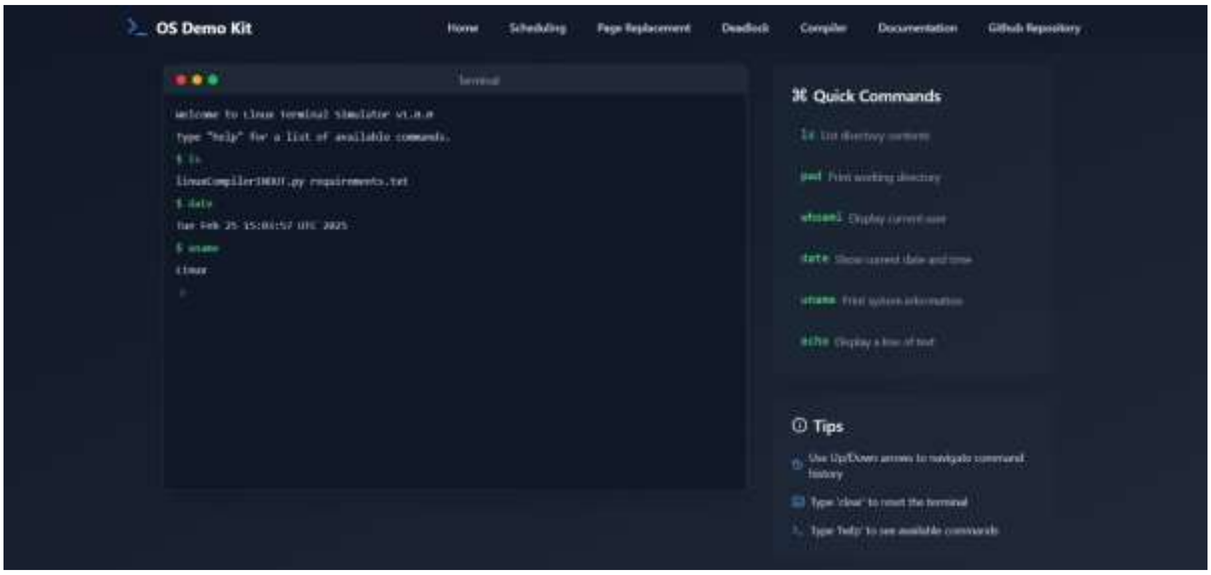
Algorithm: Least Recently Used (LRU)

Memory State Visualization

-	7	7	7	7	0	0	0	0	0	0	0	0	2	2	2	0	0	0
-	-	0	0	0	0	0	0	0	0	0	0	3	3	3	3	3	1	1
-	-	-	1	2	2	3	0	4	2	3	0	0	0	1	2	2	2	7

Performance Statistics

Fig.3. Page Replacement Algorithm



```
Welcome to Linux Terminal Simulator v1.0.0
Type "help" for a list of available commands.
$ ls
linuxCompilerINOUT.py requirements.txt
$ date
Tue Feb 25 15:01:57 UTC 2025
$ uname
Linux
>
```

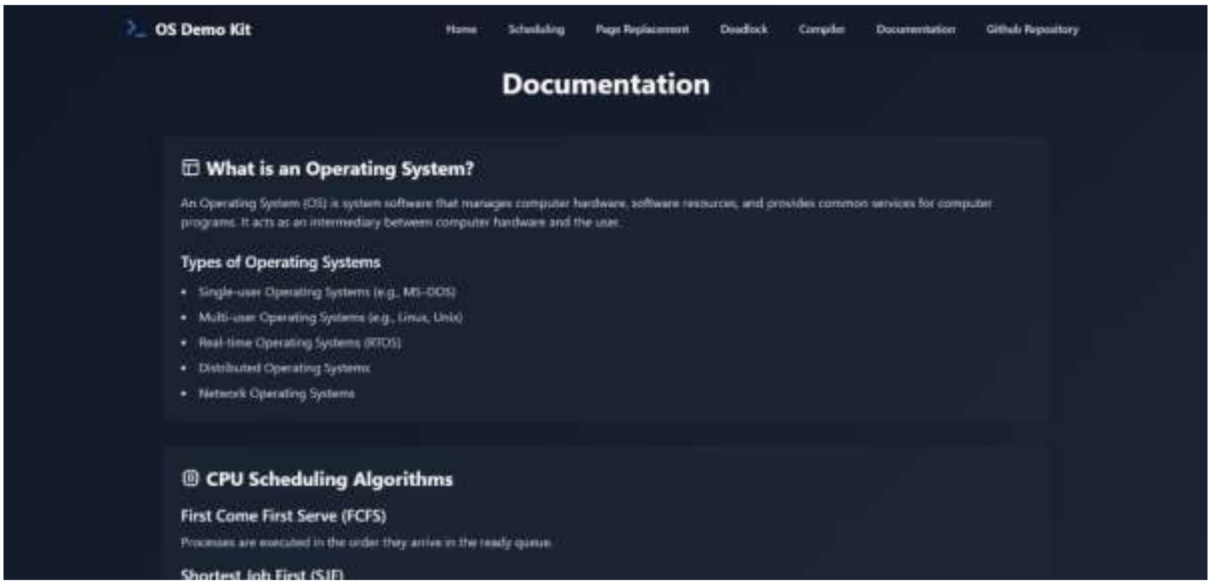
⌘ Quick Commands

- ls List directory contents
- pwd Print working directory
- whoami Display current user
- date Show current date and time
- uname Print system information
- echo Display a line of text

Tips

- Use Up/Down arrows to navigate command history
- Type 'clear' to reset the terminal
- Type 'help' to see available commands

Fig.4. Linux Terminal



Documentation

What is an Operating System?

An Operating System (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs. It acts as an intermediary between computer hardware and the user.

Types of Operating Systems

- Single-user Operating Systems (e.g., MS-DOS)
- Multi-user Operating Systems (e.g., Linux, Unix)
- Real-time Operating Systems (RTOS)
- Distributed Operating Systems
- Network Operating Systems

CPU Scheduling Algorithms

First Come First Serve (FCFS)

Processes are executed in the order they arrive in the ready queue.

Shortest Job First (SJF)

Fig.5. Documentation

Conclusion:

The **OSY DEMO KIT** bridges the gap between theory and practice in OS concepts like CPU scheduling, page replacement, and deadlock handling. With interactive simulations, real-time visualizations, and customizable parameters, it enhances learning by enabling users to experiment, analyze, and compare algorithms. Its intuitive design and cross-platform accessibility make it a valuable resource for students, educators, and researchers in computer science.

References:

[1]https://github.com/

[2].CPU Scheduling Application

[3]Scheduling Algorithms by Peter Brucker

[4] Operating System: Design and Implementation

[5] The Linux Programming Interface

[6] Linux Kernel Development By Robert Love

[7] The Art Of Unix Programming