**PROJECT AND TEAM INFORMATION**

# Project Title

Simulated Operating System Core

# Student / Team Information

|  |  |
| --- | --- |
| Team Name: SE(OS)-VI-T142  Team |  |
| **Team member 1 (Team Lead)** | Agarwal, Rishika-2021969  [Rishikaagarwal2728@gmail.com](mailto:Rishikaagarwal2728@gmail.com) |
| **Team member 2** | Kapur, Parth- 2021817  [parthkapur27@gmail.com](mailto:parthkapur27@gmail.com) |
| **Team member 3** | Negi, Rahul-2021823  [negirahul4167@gmail.com](mailto:negirahul4167@gmail.com) |
| **Team member 4** | Jalan, Akshara-2021770  [jalanakshara2004@gmail.com](mailto:jalanakshara2004@gmail.com) |

**PROPOSAL DESCRIPTION (10 pts)**

# Motivation

Understanding how an operating system works is one of the most fundamental yet complex parts of computer science education. While textbooks and theory explain the logic behind concepts like process scheduling or memory management, students often struggle to grasp how these algorithms behave in real scenarios. Most learners find it difficult to visualize the step-by-step execution of algorithms or the effect of changing parameters like burst time, priority, or memory size. This project is driven by the motivation to bridge that gap — by creating an interactive, web-based platform that brings OS algorithms to life. This is particularly important for students and developers who benefit from hands-on experience. Moreover, this project emphasizes the importance of performance optimization and resource allocation, which are critical challenges in both academic and industry-level computing. In an era of increasing demand for responsive and resource-efficient systems, gaining a deeper understanding of OS behaviour has real-world value.

**Key Benefits:**

* Visual Learning Aid
* Hands-On Development Experience
* Modular and Expandable Design
* Educational Use and Broader Applicability

# State of the Art / Current solution

Most Operating System concepts are currently taught through static diagrams, theory, and limited desktop-based simulators. While tools like Gantt chart visualizers or disk scheduling apps exist, they are often standalone, lack interactivity, and don’t cover a wide range of topics in one platform. Additionally, many aren't web-based, reducing accessibility and scalability.

Existing tools like Little Man Computer (LMC), EduOS, NachOS, and online visualizers offer insights into specific OS concepts but are often code-heavy, non-interactive, or narrowly focused. Many lack real-time interactivity and do not provide a unified learning experience.

This project addresses these gaps by offering a comprehensive, web-based, and interactive simulation platform. It integrates multiple OS functionalities into a single, user-friendly interface, making it easier for learners to visualize, compare, and understand system behaviours in real time.

# Project Goals and Milestones

Project Goals

* Design and develop an interactive web application that simulates core Operating System concepts such as process scheduling, disk scheduling, memory management, paging, file systems, and deadlock Detection.
* Implement a responsive frontend using Angular and a robust backend using Django to handle user input, simulate algorithm execution, and return visual output.
* Provide a user-friendly interface that allows learners to experiment with different algorithm parameters and visualize results step-by-step.

Project Milestones

**Initial Phase**

* Define project scope and finalize system architecture
* Set up development environment and establish frontend/backend framework

**Development Phase**

* Implement core **Process Scheduling** module with interactive visualizations
* Develop **disk scheduling** module with interactive visualizations
* Implement **paging and memory management** simulation with user-friendly UI
* Implement **Banker’s Algorithm** and **Wait-For Graph** algorithm
* Integrate all modules into a unified, web-based dashboard
* Conduct thorough **testing and debugging** for functionality and accuracy

**Final Phase**

* Perform **user testing** and gather feedback for improvements
* Optimize performance and enhance UI/UX based on feedback
* Prepare and deliver **final documentation and presentation**

Project Approach

* Define simulation scope covering four core OS topics: process scheduling, disk scheduling, memory management, and deadlock handling.
* Divide team responsibilities based on modules and expertise.
* Design a clean and modular UI using Angular with forms and interactive elements.
* Build REST APIs using Django to process input and trigger simulations.
* Implement OS algorithms in Python on the backend for accurate simulation.
* Connect Angular frontend to Django backend using HTTP requests (via Angular services).
* Use charting libraries (like Chart.js or ngx-charts) to visualize simulation results.
* Test each module independently, then integrate into a single platform.
* Optimize for responsiveness, input validation, and user experience.
* Document the project thoroughly and prepare for deployment/demo

# System Architecture

1. Client Layer (Frontend – Angular)

* Accepts user input: algorithm type, parameters (e.g., burst time, page size, etc.)
* Sends requests to the backend via REST API
* Displays results using tables and charts (e.g., Gantt charts, memory maps)
* Built with Angular components and services

2. Application Layer (Backend – Django)

* Exposes RESTful APIs using Django REST Framework
* Receives input data from the frontend
* Invokes appropriate simulation logic based on the input
* Sends processed results (e.g., scheduling order, page faults) back to the frontend in JSON format

3. Simulation Logic Layer (Python Modules)

* Contains all core algorithm implementations:  
  • Process Scheduling (FCFS, SJF, RR, Priority)  
  • Disk Scheduling (SSTF, SCAN, C-SCAN)  
  • Memory Management (Paging Algorithm)  
  • Deadlock Detection(Banker’s Algorithm, Wait-For Graph)
* Each algorithm is implemented as a separate module or function

4. Technologies Involved

* Frontend: Angular, HTML, CSS
* Backend: Python, Django, Django REST Framework

# Project Outcome / Deliverables

The outcome of this project is a fully functional, web-based simulation platform that allows users to interactively explore and understand fundamental Operating System concepts. By simulating key topics such as process scheduling, disk scheduling, memory management, and deadlock handling, the platform provides a visual and intuitive learning experience. The system enables users to input custom parameters, observe real-time algorithm behavior, and interpret results through dynamic charts and tables. In addition to strengthening our own understanding of OS algorithms, the project serves as a practical educational tool for students and educators, demonstrating how theoretical concepts are applied in real-world scenarios using modern web development frameworks.

# References

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.
2. Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems (4th ed.). Pearson Education.
3. Django Software Foundation. (2024). Django Documentation. Retrieved from <https://docs.djangoproject.com/>
4. Angular Team. (2024). Angular - The Modern Web Developer's Platform. Retrieved from <https://angular.io/>
5. MySQL Documentation. (2024). MySQL 8.0 Reference Manual. Oracle. Retrieved from <https://dev.mysql.com/doc/>
6. W3Schools. (2024). HTML, CSS, JavaScript Tutorials. Retrieved from <https://www.w3schools.com/>
7. Chart.js Contributors. (2024). Chart.js Documentation. Retrieved from https://www.chartjs.org/docs/latest/
8. GeeksforGeeks. (2024). Operating System Tutorials and Algorithm Implementations. Retrieved from https://www.geeksforgeeks.org/operating-systems/
9. TutorialsPoint. (2024). Operating System Concepts and Scheduling Algorithms. Retrieved from https://www.tutorialspoint.com/operating\_system/index.htm
10. MDN Web Docs. (2024). HTML, CSS, JavaScript Reference. Mozilla. Retrieved from <https://developer.mozilla.org/>