

**LOVELY PROFESSIONAL UNIVERSITY**

**INVENTION DISCLOSURE FORM**

1. **TITLE: Secure File Management System**

# FACULTY: Gagandeep Kaur

1. **STUDENT(S):**

|  |  |
| --- | --- |
| A. Full name | SURYA TEJA VODAPALLY |
| Roll Number | 13 |
| Registration number | 12303954 |
| B. Full name | BEHARA PRUDHVI |
| Roll Number | 31 |
| Registration number | 12318633 |
| C. Full name | B GAGAN KUMAR REDDY |
| Roll Number | 50 |
| UID/Registration number | 12303967 |

1. Github link

**Project Overview: Intelligent CPU Scheduler Simulator**

The **Intelligent CPU Scheduler Simulator** is a Python-based application designed to simulate various CPU scheduling algorithms, including **First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), and Priority Scheduling**. This simulator provides **real-time visualizations** of scheduling processes through Gantt charts and calculates essential performance metrics such as **average waiting time (AWT), turnaround time (TAT), and response time**.

The system is built with a **graphical user interface (GUI) using Tkinter** to allow users to input process details like **arrival time, burst time, and priority**, and dynamically observe how different scheduling strategies affect process execution. The simulator aims to enhance the understanding of CPU scheduling concepts by providing interactive and visual representations.

# 2. Module-Wise Breakdown: Intelligent CPU Scheduler Simulator

The project is divided into multiple modules, each handling a specific function of the CPU scheduling simulation:

# User Input Module

* + Allows users to enter process details such as **process ID, arrival time, burst time, and priority (if applicable)**.
  + Supports batch input from a file or manual entry.
  + Provides error handling for invalid inputs.

# Scheduling Algorithms Module

* + Implements the following CPU scheduling algorithms:
    - **FCFS (First Come, First Serve)**
    - **SJF (Shortest Job First) – Preemptive & Non-Preemptive**
    - **Round Robin (RR) – Time Quantum Selection**
    - **Priority Scheduling – Preemptive & Non-Preemptive**
  + Dynamically updates and executes the scheduling logic.

# Gantt Chart Visualization Module

* + Generates **real-time Gantt charts** to visualize the execution order of processes.
  + Uses **Matplotlib** to display execution timelines.
  + Animates process execution in real-time for better understanding.

# Performance Metrics Module

* + Calculates and displays:
    - **Average Waiting Time (AWT)**
    - **Turnaround Time (TAT)**
    - **Throughput and CPU Utilization**
  + Provides comparisons between different scheduling algorithms.

# Graphical User Interface (GUI) Module

* + Built using **Tkinter** for user interaction.

# Allows users to select scheduling algorithms, enter process details, and view Gantt charts.

* + Displays real-time performance metrics in an intuitive layout.

# Data Handling Module

* + Stores scheduling results for later analysis.
  + Allows users to **export results (CSV or JSON)** for further study.

# Supports importing process data from external files.

1. **Functionalities of Intelligent CPU Scheduler Simulator**

The simulator offers the following key functionalities:

# Process Management

* + **Add/Delete/Modify** processes dynamically.
  + **Load processes from a file** for batch execution.
  + Supports both **preemptive and non- preemptive scheduling**.

# Scheduling Execution

* + Executes **FCFS, SJF, Round Robin, and Priority Scheduling** based on user

selection.

* + Provides **step-by-step execution** to demonstrate how scheduling decisions are made.

# Real-Time Visualization

* + Displays **dynamic Gantt charts** to illustrate CPU execution order.
  + Highlights **waiting times, response times, and turnaround times** in graphical form.

# Performance Analysis

* + Computes:
    - **Average Waiting Time (AWT)**
    - **Turnaround Time (TAT)**
    - **Response Time**
    - **CPU Utilization**
    - **Throughput**
  + Compares multiple scheduling algorithms to determine **efficiency and fairness**.

# Interactive GUI

* + **User-friendly interface** for entering and managing processes.
  + Dropdown menus to **switch between scheduling algorithms**.

# Visual performance comparison graphs.

1. **Technology Used**

The project utilizes various technologies and libraries to ensure efficient simulation and visualization:

# Programming Language

* + **Python 3.11**: Core development language.

# Virtual Environment

* + **venv**: Used for dependency management.

# Libraries & Packages

* + **Tkinter** → Provides an interactive GUI.
  + **Matplotlib** → Generates Gantt charts for scheduling visualization.
  + **Pandas** → Handles process data storage and manipulation.
  + **Numpy** → Optimizes calculations for scheduling algorithms.
  + **Time & Threading** → Simulates real- time process execution.

# Environment & Dependency Management

* + **Python Virtual Environment (venv)** to manage dependencies.

# Data Flow Diagram

1. **Overall System Flowchart**

scss CopyEdit

┌────────────────────────

┐

│ User Input │

└────────────────────────

┘

↓

┌────────────────────────

┐

│ Scheduling Algorithm │

│ (FCFS, SJF, RR, Priority) │

└────────────────────────

┘

↓

┌────────────────────────

┐

│ Gantt Chart Display │

└────────────────────────

┘

↓

┌────────────────────────

┐

│ Performance Metrics │

│ (AWT, TAT, CPU Utilization) │

└────────────────────────

┘

↓

┌────────────────────────

┐

│ Report Generation │

└────────────────────────

┘

# Detailed Process Flow

mathematica CopyEdit

┌────────────────────────

────────┐

│ Start Simulation │

└────────────────────────

────────┘

↓

┌────────────────────────

────────┐

│ Enter Process Data (GUI) │

└────────────────────────

────────┘

↓

┌────────────────────────

────────┐

│ Select Scheduling Algorithm │

└────────────────────────

────────┘

↓

┌────────────────────────

────────┐

│ Compute Scheduling Order │

└────────────────────────

────────┘

↓

┌────────────────────────

────────┐

│ Generate Gantt Chart │

└────────────────────────

────────┘

↓

┌────────────────────────

────────┐

│ Display Performance Metrics │

└────────────────────────

────────┘

↓

┌────────────────────────

────────┐

│ Save Report / Export Data │

└────────────────────────

────────┘

# Conclusion & Future Scope Conclusion

The **Intelligent CPU Scheduler Simulator** provides a robust and interactive way to **visualize, analyze, and compare CPU scheduling algorithms**. By implementing **real-time Gantt charts and performance metrics**, the simulator offers an educational and practical tool for understanding process scheduling in operating systems.

The simulator helps users **experiment with different scheduling techniques, compare efficiency, and make informed decisions** regarding process management.

# Future Scope

⬛ **AI-Based Scheduling** → Implement AI algorithms to **dynamically select the best scheduling method** based on system load.

⬛ **Machine Learning Integration** → Train models to predict **optimal scheduling strategies** based on historical data.

⬛ **Cloud-Based Deployment** → Provide a **web-based** version of the simulator for remote access.

⬛ **Support for Multi-Core Processors** → Extend the system to **simulate multi-core CPU scheduling**.

⬛ **Mobile Application** → Develop an

Android/iOS app with similar scheduling functionalities.

⬛ **Real-Time OS Scheduling** → Expand the simulator to **model real-time operating system scheduling policies**.

# References

1. **CPU Scheduling in Operating Systems**
   * [OS Concepts Book](https://en.wikipedia.org/wiki/Scheduling_(computing))
2. **Python Matplotlib for Gantt Charts** – Matplotlib Docs
3. **Tkinter GUI Programming** – [Python](https://docs.python.org/3/library/tkinter.html) [Official Docs](https://docs.python.org/3/library/tkinter.html)

# Operating Systems: Three Easy Pieces

* + OSTEP

1. **Pandas for Data Handling** – Pandas Docs