LOVELY PROFESSIONAL UNIVERSITY

Academic Task-2

School of Computer Science and Engineering	Faculty of Technology and Sciences
Course Code: CSE316	Course Title: Operating Systems
Term: 24252	Max. Marks:30

Instructions for Assignment Submission

1. This Project is a compulsory CA component.

2. It is a group activity but submission on UMS will be individual by each student.

3. The submission mode is **Online** only. The student has to upload the complete project report on or before the last date on UMS only. No submission via e-mail, pen-drive or any media will be accepted.

4. Non-submission of project/ report on **UMS** till the last date will result in **ZERO** marks.

5. The student is supposed to code the project on his/her own. If it is discovered at any stage that the student has used unfair means like copying from peers or copy-pasting the code taken from the internet, AI etc. **ZERO** marks will be awarded to the student.

6. The student who will share his/her project solution with other students, (either in the same section or a different section) will also get **ZERO** marks.

Phase 1: Problem Assignment

Guidelines For Students:

Step 1: AI-Guided Development

AI Guidance:

- Students will submit the problem statement to the AI system.
- AI will analyze the problem and provide a **detailed explanation** of the project statement along with its breakdown.

The prompt that will be asked by a student to AI:

I have been assigned the following problem statement:

"Paste your Problem Statement Here."

Provide the following details to help me implement this project:

1. *Project Overview: Explain the project's goals, expected outcomes, and scope.*
2. *Module-Wise Breakdown: Divide the project into 3 separate modules (e.g., GUI, ML, data visualization), explaining their purposes and roles.*
3. *Functionalities: List key features for each module, with examples for clarity.*
4. *Technology Recommendations: Suggest programming languages, libraries, and tools to use.*
5. *Execution Plan: Provide a step-by-step implementation guide with tips for efficiency.*

Implementation Language: Any as per your project requirement

Step 2: Revision Tracking on GitHub

GitHub Workflow:

- Students will:
  - Create a repository for the project.
  - Maintain at least 7 revisions and push changes regularly with clear commit messages.
  - Use branches for feature implementation and merge them into the main branch after testing.

Step 3: Report Format

1. Project Overview
2. Module-Wise Breakdown
3. Functionalities
4. Technology Used
   - Programming Languages:
   - Libraries and Tools:
   - Other Tools: [e.g., GitHub for version control]
5. Flow Diagram
6. Revision Tracking on GitHub
   - Repository Name: [Insert Repository Name]
   - GitHub Link: [Insert Link]
7. Conclusion and Future Scope

8. References

Appendix

A. AI-Generated Project Elaboration/Breakdown Report

[Paste the AI-generated breakdown of the project in detail]

B. Problem Statement: [Paste Problem Statement]

C. Solution/Code:

[Paste complete code of entire project here]

Phase 2: Project Evaluation (30 marks)

The entire evaluation has been divided into 2 steps:

Step 1: AI Evaluation (15 marks)

The student will upload the entire project as per the shared report on UMS within the deadline. The faculty members will use AI to evaluate the project submitted by the student. This AI evaluation will be out of 15 marks.

Step 2: Presentation and Viva (15 marks)

The faculty members will take the presentation and viva of the students in their respective classes and will evaluate the students as per the shared rubrics.

Rubric for Presentation and Viva

Criterion	**Details**	**Marks**
Understanding of Project and Question response	Depth of knowledge about the problem statement, implementation, challenges, and correct question response.	**10**
- Excellent	Demonstrates thorough understanding with clear and concise answers.	7
- Good	Good understanding but some minor gaps.	5
- Needs Improvement	Limited understanding and vague answers.	3
Problem-Solving Approach	Innovative solutions and effective handling of challenges.	**3**

Criterion	Details	Marks
- Excellent	Showcases innovation and robust problem-solving skills.	3
- Good	Reasonable problem-solving with minor shortcomings.	2
- Needs Improvement	Basic or ineffective problem-solving approach.	1
Communication Skills	Clarity and confidence in explaining the project.	**2**
- Excellent	Articulates ideas effectively with high confidence.	2
- Good	Reasonably clear explanations with some hesitation.	1
- Needs Improvement	Difficulty in articulating thoughts or lack of confidence.	0

Project Statements

1. Intelligent CPU Scheduler Simulator

Description: Develop a simulator for CPU scheduling algorithms (FCFS, SJF, Round Robin, Priority Scheduling) with real-time visualizations. The simulator should allow users to input processes with arrival times, burst times, and priorities and visualize Gantt charts and performance metrics like average waiting time and turnaround time.

2. Problem: Dynamic Memory Management Visualizer

Description: Build a tool to simulate and visualize memory management techniques like paging, segmentation, and virtual memory. The system should handle user-defined inputs for memory allocation, page faults, and replacement algorithms (FIFO, LRU).

3. Problem: Inter-Process Communication (IPC) Debugger

Description: Design a debugging tool for inter-process communication methods (pipes, message queues, shared memory) to help developers identify issues in synchronization and data sharing between processes. Include a GUI to simulate data transfer and highlight potential bottlenecks or deadlocks.

4. Deadlock Prevention and Recovery Toolkit

Description: Create a toolkit that detects, prevents, and recovers from deadlocks in real-time. The system should implement algorithms like Banker's Algorithm and display

resource allocation graphs for clarity. Include a feature for simulating deadlock scenarios with custom user inputs.

5. Secure File Management System

Description: Develop a secure file management system that incorporates authentication mechanisms (password-based, two-factor), protection measures (access control, encryption), and detection of common security threats (buffer overflow, malware). Users should be able to perform file operations like read, write, share, and view metadata securely.

6. Real-Time Multi-threaded Application Simulator

Description: Develop a simulator to demonstrate multithreading models (e.g., Many-to-One, One-to-Many, Many-to-Many) and thread synchronization using semaphores and monitors. The simulator should visualize thread states and interactions, providing insights into thread management and CPU scheduling in multi-threaded environments.

7. Virtual Memory Optimization Challenge

Description: Create a virtual memory management tool that visualizes the behavior of paging and segmentation, including page faults and demand paging. Allow users to experiment with custom inputs for memory allocation, simulate memory fragmentation, and evaluate page replacement algorithms like LRU and Optimal.

8. File System Recovery and Optimization Tool

Description: Design a tool for recovering and optimizing file systems. Implement methods for free-space management, directory structures, and file access mechanisms. Simulate real-world scenarios like disk crashes and provide recovery techniques while optimizing file read/write times.

9. Security Vulnerability Detection Framework

Description: Develop a framework to detect and mitigate security vulnerabilities in operating systems, such as buffer overflows, trapdoors, and cache poisoning. The framework should simulate attacks, provide real-time detection alerts, and suggest recovery or prevention measures.

10. Advanced Disk Scheduling Simulator

Description: Build a disk scheduling simulator to visualize algorithms like FCFS, SSTF, SCAN, and C-SCAN. Users should be able to input custom disk access requests and see

how the algorithms minimize seek time. Include performance metrics like average seek time and system throughput.

11. Adaptive Resource Allocation in Multiprogramming Systems

Description: Develop a system that dynamically adjusts resource allocation among multiple programs to optimize CPU and memory utilization. The solution should monitor system performance and reallocate resources in real-time to prevent bottlenecks.

12. User-Friendly System Call Interface for Enhanced Security

Description: Design an intuitive interface for system calls that enhances security by preventing unauthorized access. The interface should include authentication mechanisms and provide detailed logs of system call usage.

13. Real-Time Process Monitoring Dashboard

Description: Create a graphical dashboard that displays real-time information about process states, CPU usage, and memory consumption. The tool should allow administrators to manage processes efficiently and identify potential issues promptly.

14. Energy-Efficient CPU Scheduling Algorithm

Description: Develop a CPU scheduling algorithm that minimizes energy consumption without compromising performance. The algorithm should be suitable for mobile and embedded systems where power efficiency is critical.

15. Scalable Thread Management Library

Description: Design a thread management library that supports efficient creation, synchronization, and termination of threads. The library should be scalable to handle thousands of threads concurrently, suitable for high-performance computing applications.

16. Automated Deadlock Detection Tool

Description: Develop a tool that automatically detects potential deadlocks in system processes. The tool should analyze process dependencies and resource allocation to identify circular wait conditions and suggest resolution strategies.

17. Secure Authentication Module for Operating Systems

Description: Create a robust authentication module that integrates with existing operating systems to enhance security. The module should support multi-factor authentication and protect against common vulnerabilities like buffer overflows and trapdoors.

18. Efficient Page Replacement Algorithm Simulator

Description: Design a simulator that allows users to test and compare different page replacement algorithms (e.g., FIFO, LRU, Optimal). The simulator should provide visualizations and performance metrics to aid in understanding algorithm efficiency.

19. Distributed File System with Fault Tolerance

Description: Develop a distributed file system that ensures data availability and integrity across multiple nodes. The system should incorporate fault tolerance mechanisms to handle node failures gracefully.

20. Comprehensive Inter-Process Communication Framework

Description: Create a framework that facilitates efficient inter-process communication (IPC) using various methods such as pipes, message queues, and shared memory. The framework should include security features to prevent unauthorized data access.

21. Dynamic Load Balancing in Multiprocessor Systems

Description: Develop an algorithm that dynamically distributes workloads across multiple processors to optimize performance and resource utilization. The solution should adapt to varying process loads and system states.

22. User-Friendly System Call Interface for Enhanced Security

Description: Design an intuitive interface for system calls that enhances security by preventing unauthorized access. The interface should include authentication mechanisms and provide detailed logs of system call usage.

23. Problem: Real-Time Process Monitoring Dashboard

Description: Create a graphical dashboard that displays real-time information about process states, CPU usage, and memory consumption. The tool should allow administrators to manage processes efficiently and identify potential issues promptly.

24. Energy-Efficient CPU Scheduling Algorithm

Description: Develop a CPU scheduling algorithm that minimizes energy consumption without compromising performance. The algorithm should be suitable for mobile and embedded systems where power efficiency is critical.

25. Scalable Thread Management Library

Description: Design a thread management library that supports efficient creation, synchronization, and termination of threads. The library should be scalable to handle thousands of threads concurrently, suitable for high-performance computing applications.

26. Automated Deadlock Detection Tool

Description: Develop a tool that automatically detects potential deadlocks in system processes. The tool should analyze process dependencies and resource allocation to identify circular wait conditions and suggest resolution strategies.

27. Secure Authentication Module for Operating Systems

Description: Create a robust authentication module that integrates with existing operating systems to enhance security. The module should support multi-factor authentication and protect against common vulnerabilities like buffer overflows and trapdoors.

28. Efficient Page Replacement Algorithm Simulator

Description: Design a simulator that allows users to test and compare different page replacement algorithms (e.g., FIFO, LRU, Optimal). The simulator should provide visualizations and performance metrics to aid in understanding algorithm efficiency.

29. Distributed File System with Fault Tolerance

Description: Develop a distributed file system that ensures data availability and integrity across multiple nodes. The system should incorporate fault tolerance mechanisms to handle node failures gracefully.

30. Comprehensive Inter-Process Communication Framework

Description: Create a framework that facilitates efficient inter-process communication (IPC) using various methods such as pipes, message queues, and shared memory. The framework should include security features to prevent unauthorized data access.

31. AI-powered performance Analyzer for OS Processes

Description: Build a tool leveraging AI to analyze the performance of system processes in real-time. Highlight bottlenecks, suggest optimizations, and forecast future resource

requirements.

32. Graphical Simulator for Resource Allocation Graphs
Description: Create a visual tool to simulate resource allocation graphs and analyze deadlock scenarios interactively.

33. Real-Time OS Security Event Logger
Description: Design a logger that monitors and records OS-level security events, providing insights into potential vulnerabilities.

34. Adaptive OS Scheduler for Real-Time Systems
Description: Create an adaptive OS scheduler that dynamically adjusts priorities in real-time systems based on workload changes and deadlines.

35. Real-Time Process Synchronization Analyzer
Description: Create a tool to analyze and resolve process synchronization issues in multi-threaded applications.

36. AI-Powered Directory Management System
Description: Build an AI-based directory management system that categorizes and organizes files efficiently.

37. Efficient Garbage Collection in OS
Description: Design an optimized garbage collection mechanism for memory management in modern OS environments.

38. AI-Powered Deadlock Detection System
Description: Design an AI-driven system to predict, detect, and resolve deadlocks in real-time.

39. Real-Time Memory Allocation Tracker
Description: Build a tool that visualizes memory allocation in real-time, showcasing paging and segmentation.

40. Secure Inter-process Communication Framework
Description: Develop a secure inter-process communication framework to prevent unauthorized access.

41. AI-Enhanced System Call Optimization

Description: Implement AI techniques to optimize system call processing and reduce latency in OS.

42. AI-Based Disk Scheduling Algorithms

Description: Implement and compare AI-based approaches to disk scheduling in operating systems.