A

SYNOPSIS REPORT

On

Resource Allocation And Deadlock Detection System

Submitted in partial fulfilment of the requirements of the degree of

BACHELOR OF COMPUTER APPLICATIONS

ITM UNIVERSITY GWALIOR • MP • INDIA "CELEBRATING DREAMS"

Submitted by--

Name	*Roll no.*	*Branch/Sem*
Aditya Sharma	**BCAN1CA24013**	**BCA/2nd**
Abhi YADAV	**BCAN1CA24002**	**BCA/2nd**
Anirudh kushwah	**BCAN1CA24023**	**BCA/2nd**

Department of Computer Science and Applications

School of Engineering and Technology

ITM University Gwalior, Madhya Pradesh

JANUARY 2025

RESOURCES ALLOCATION AND DEADLOCK DETECTION SYSTEM

1. Problem Statement

In modern computing environments, efficient resource allocation is crucial to ensure smooth execution of processes. However, improper resource management can lead to deadlocks, where processes wait indefinitely for resources held by others. This results in system inefficiencies, increased response time, and potential system crashes. Deadlocks can occur in various environments such as multi-threaded applications, distributed systems, database management, and operating systems.

The primary challenge in resource management is the allocation and deallocation of resources in a manner that prevents deadlocks while ensuring optimal utilization. Without a proper system in place, multiple processes may request access to limited resources, leading to circular wait conditions. Traditional operating systems and software solutions use preemptive or non-preemptive techniques to handle such scenarios, but these methods have their drawbacks.

This project aims to implement an efficient Resource Management and Deadlock Detection System in C, which monitors system resources, detects potential deadlocks, and resolves them by following effective allocation algorithms. The system will simulate resource allocation scenarios to study how deadlocks arise and provide an automated solution to detect and handle them.

Objective, Scope, and Limitations

Objective:

1. To develop a system that efficiently allocates resources to processes while preventing deadlocks.
2. To implement deadlock detection mechanisms using algorithms like the Banker’s Algorithm and Wait-For Graph.
3. To provide a user-friendly interface for tracking resource allocation and process status.

Scope:

1. This system will be applicable in operating systems, distributed computing, and real-time applications.
2. It will help optimize resource utilization, reduce system downtime, and enhance process efficiency.
3. The project can be extended to cover multi-threaded and network-based resource allocation scenarios.

Limitations:

1. The project is limited to single-system environments and does not include real-time cloud-based resource allocation.
2. The deadlock handling strategies implemented may not be suitable for large-scale industrial applications without modifications.
3. The simulation will be based on predefined test cases and may not cover all possible real-world scenarios.

Methodology

The project follows a structured approach:

1. **Requirement Analysis:** Identify essential resources and processes that require allocation in a simulated environment.
2. **Design Phase:** Develop a resource allocation model, create UML diagrams, and outline process flow.
3. **Implementation:** Write C programs to simulate resource management and deadlock detection using algorithms like Banker's Algorithm, Resource Allocation Graph, and Wait-For Graph.
4. **Testing:** Verify correct resource allocation and detection of deadlocks under various conditions, including worst-case scenarios. Unit testing and integration testing will be performed.
5. **Optimization:** Improve algorithm efficiency by analyzing time complexity and reducing redundant computations.
6. **Finalization:** Document the project, prepare a user manual, and evaluate performance based on test results.

Additionally, the system will provide real-time monitoring and logging features to track resource requests, allocations, and potential deadlocks. A graphical user interface (GUI) may also be considered for better visualization of deadlock occurrences and system performance.

Hardware & Software to be Used

Hardware:

- Intel-based processor (i3 or above)
- Minimum 4GB RAM
- Storage: 10GB free space

Software:

- Operating System: Windows/Linux
- Programming Language: C
- IDE: Code::Blocks/Dev-C++/GCC Compiler

Application and Future Scope

Application:

- Used in operating systems for resource scheduling and process management.
- Helps in database transaction management to avoid deadlocks.
- Useful in cloud computing for optimizing resource allocation.

Future Scope:

- Extend the system to support dynamic resource allocation.

- Implement AI-based predictive resource management.
- Adapt the system for multi-threaded and distributed environments.

References/Bibliography

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. Wiley.
2. Stallings, W. (2014). Operating Systems: Internals and Design Principles. Pearson.
3. Online resources: GeeksforGeeks, Stack Overflow, and official C programming documentation.