# PROJECT AND TEAM INFORMATION

## Project Title

*(Try to choose a catchy title. Max 20 words).*

**Deadlock Resolution Simulation**

Student / Team Information

|  |  |
| --- | --- |
| *Team Name: Team #* | ***Task Titans*** |
| **Team member 1 (Team Lead)**  *(Last Name, name: student ID: email, picture):* | *Kumar, Vrijendra – 220112319* [*Vrijendrakumar393@gmail.com*](mailto:Vrijendrakumar393@gmail.com) |
| **Team member 2**  *(Last Name, name: student ID: email, picture):* | *Sharma, Shiv charan – 220111089* [*Shivcharan.gehu@gmail.com*](mailto:Shivcharan.gehu@gmail.com) |

|  |  |
| --- | --- |
| *Team Name: Team #* | *Task Titans* |
| **Team member 1 (Team Lead)**  *(Last Name, name: student ID: email, picture):* | *Chauhan, Arpit Kumar –230115096* [*Kumarchauhanarpit@gmail.com*](mailto:Kumarchauhanarpit@gmail.com) |
| **Team member 2**  *(Last Name, name: student ID: email, picture):* | *Chauhan, Varnit – 22011402* [*varnitchauhan97@gmail.com*](mailto:varnitchauhan97@gmail.com) |

# PROPOSAL DESCRIPTION (10 pts)

## Motivation (1 pt)

*(Describe the problem you want to solve and why it is important. Max 300 words).*

Deadlock is a critical issue in operating systems and distributed computing, where multiple processes compete for shared resources, leading to a state where no process can proceed. This situation significantly impacts system performance, causing delays, resource wastage, and potential failures in critical applications such as banking, airline reservations, and real-time embedded systems.

This project aims to **simulate deadlock resolution techniques** to better understand and evaluate their effectiveness in real-world scenarios. By creating a simulation, we can analyze how deadlocks occur, detect them efficiently, and apply resolution strategies such as **resource preemption, process rollback, and termination**. Understanding deadlock resolution is crucial because poorly handled deadlocks can lead to system crashes, security vulnerabilities, and degraded user experience.

By implementing algorithms like **Banker’s Algorithm, Wait-for Graph Analysis, and Resource Allocation Graph Detection**, we can compare their efficiency and effectiveness. The insights gained from this project can help optimize operating system schedulers, database management systems, and distributed networks to handle deadlocks more efficiently.

This simulation will provide a **practical demonstration of deadlock management techniques**, making it easier for students, researchers, and engineers to grasp the complexities of concurrent system operations and resource management.

## State of the Art / Current solution (1 pt)

*(Describe how the problem is solved today (if it is). Max 200 words).*

Deadlock resolution is handled using **detection, prevention, and recovery mechanisms** in modern computing systems.

1. **Deadlock Prevention**: Systems use strategies like **resource ordering, hold-and-wait avoidance, and circular wait prevention** to stop deadlocks before they occur. **Banker’s Algorithm** is widely used in OS and databases for this purpose.
2. **Deadlock Detection**: Many systems implement **Resource Allocation Graphs (RAG)** and **Wait-for Graphs** to identify cycles that indicate deadlock. **Timeout-based detection** is common in networks and distributed systems.
3. **Deadlock Recovery**: If a deadlock occurs, systems apply **preemption (forcefully reclaiming resources), rollback (restoring processes to a safe state), or process termination**.

## Project Goals and Milestones (2 pts)

*(Describe the project general goals. Include initial milestones as well any other milestones. Max 300 words).*

*Project Goals:*

* Develop a simulation model for deadlock occurrence, detection, and resolution.
* Implement deadlock detection algorithms such as Resource Allocation Graph (RAG).
* Apply resolution strategies like preemption, rollback, and process termination to break deadlocks.
* Compare the efficiency of different detection and resolution methods.
* Provide a user-friendly interface or visualization for understanding deadlock scenarios.

*Project Milestones:*

1. Phase 1: Research & Design
   * Study deadlock scenarios and existing resolution techniques.
   * Design a process-resource interaction model.
2. Phase 2: Implementation
   * Develop deadlock detection using graph-based models.
   * Implement resolution techniques (preemption, rollback, termination).
3. Phase 3: Testing & Optimization
   * Simulate different deadlock cases and analyze resolution efficiency.
   * Optimize algorithms for faster detection and minimal resource loss.
4. Phase 4: Finalization & Documentation
   * Prepare a detailed report, performance analysis, and conclusions.
   * Demonstrate the simulation results.

## 

## Project Approach (3 pts)

*(Describe how you plan to articulate and design a solution. Including platforms and technologies that you will use. Max 300 words).*

**Project Approach**

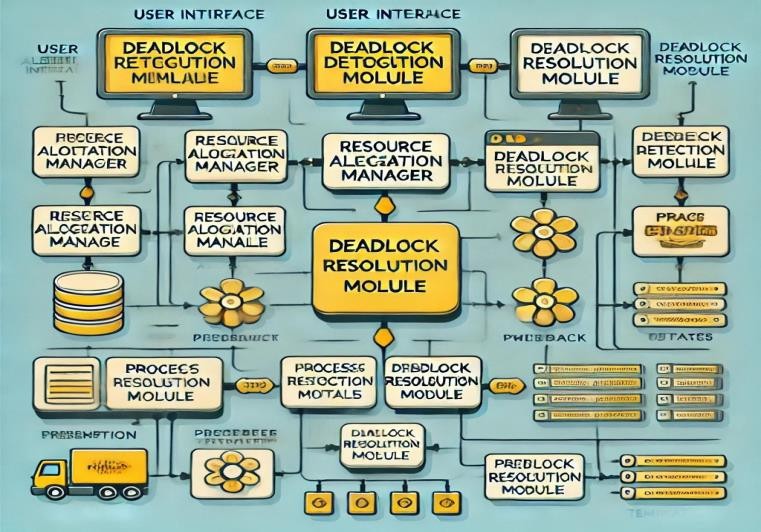
The project will follow a structured approach to simulate deadlock resolution. First, a **process-resource interaction model** will be designed, representing multiple processes requesting and releasing shared resources. This model will be implemented using **Python**, leveraging **multithreading (pthreads in Python threading module)** to simulate concurrent execution.

For **deadlock detection**, algorithms like **Resource Allocation Graph (RAG) and Wait-for Graph (WFG)** will be implemented. If a circular wait is detected, the system will apply resolution techniques such as **preemption (forcefully reclaiming resources), rollback (restoring process states), or termination (killing one or more processes to break the cycle)**.

The simulation will include **a command-line interface or GUI using Tkinter (Python)** for user interaction. Data visualization tools like **NetworkX (Python)** will be used to represent the resource allocation graph dynamically. The system will be tested under different deadlock scenarios to analyze **algorithm efficiency and resource management**.

## System Architecture (High Level Diagram) (2 pts)

*(Provide an overview of the system, identifying its main components and interfaces in the form of a diagram using a tool of your choice).*



## Project Outcome / Deliverables (1 pts)

*(Describe what are the outcomes / deliverables of the project. Max 200 words).*

The primary deliverables of this project include:

1. **Deadlock Simulation System**: A working simulation model that demonstrates deadlock occurrence, detection, and resolution strategies in a multi-process environment.
2. **Implementation of Algorithms**: Code implementing **deadlock detection algorithms** like **Resource Allocation Graph (RAG)** and **Wait-for Graph (WFG)**, along with **preemption, rollback, and process termination** strategies for resolution.
3. **Performance Analysis**: A report that compares the efficiency and effectiveness of the implemented deadlock detection and resolution techniques, including **time complexity and resource utilization**.
4. **User Interface/Visualization**: A simple interface (command-line or GUI) that visualizes resource allocation, process requests, and deadlock states using tools like **NetworkX** (Python) for graph representation.
5. **Documentation**: Detailed documentation of the design, algorithms, results, and analysis, including any limitations and potential improvements.

## Assumptions

*(Describe the assumptions ( if any ) you are making to solve the problem. Max 100 words )*

**Assumptions**

To solve the deadlock resolution problem, we assume that:

1. **Finite Resources**: Resources are limited and allocated to processes on a first-come, first-served basis.
2. **Predefined Process Requests**: Processes make resource requests and release them at specified intervals.
3. **No External Failures**: The system operates in an ideal environment without external failures, such as hardware malfunctions or network issues.
4. **Simple Deadlock Scenarios**: The model assumes standard deadlock scenarios without complex dependencies, focusing on fundamental detection and resolution techniques.

.

## References

*(Provide a list of resources or references you utilized for the completion of this delivery. You may provide links).*

1. Stallings, W. (2018). *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.
   * Includes detailed discussions on deadlock prevention algorithms and resource allocation techniques.
2. *NetworkX Documentation* (n.d.). Retrieved from <https://networkx.github.io/>
   * A Python library used for creating and manipulating complex networks and graphs, useful for simulating resource allocation graphs.
3. *Python threading module documentation*. Retrieved from [https://docs.python.org/3/library/threadi](https://docs.python.org/3/library/threading.html) [ng.html](https://docs.python.org/3/library/threading.html)
   * Used for implementing multithreading in Python to simulate concurrent processes in the project.
4. *Pthreads - POSIX threads library* (n.d.). Retrieved from [https://pubs.opengroup.org/onlinepubs/9](https://pubs.opengroup.org/onlinepubs/9699919799/) [699919799/](https://pubs.opengroup.org/onlinepubs/9699919799/)
   * Documentation for the POSIX threads library, used for multithreading in C.