Project Proposal & Conceptual Design

1) Problem statement:

- **Core Issue:**

The inability of modern radar systems to quickly distinguish friends from foes in the fog of war is a critical vulnerability, leading to delayed responses, security breaches, and friendly fire. Our solution is an efficient algorithmic approach to contact classification and management.

- **Project Overview:**

Develop a C++ simulation program that models a modern radar system capable of automatically detecting, tracking, and classifying aerial contacts as either friendly or unknown entities. The system will utilize appropriate data structures to efficiently manage and process real-time contact data while implementing Identification Friend-or-Foe (IFF) logic.

- **Key Objectives:**
  To simulate a dynamic airspace with multiple contacts appearing, moving, and disappearing.
  To implement a reliable IFF logic for classifying contacts.
  To use efficient data structures for storing, searching, and updating contact information.
  To provide a clear, real-time textual display of the airspace status and threat alerts.

2) Key Data Structures to be Implemented:

Arrays / Lists – to store detected object positions and metadata.
• **Queues** – to manage incoming sensor readings in order (FIFO) for processing.
• **Structs/Classes** – class containing ID, position (x, y), velocity, heading, classification, and threat level.
• **Linked List / Dynamic Containers** – for frequent insertions/deletions, or std::vector for cache-friendly iteration.
• **Hash Tables** – for O(1) lookup of objects by ID.
• **Priority Queues** – for threat prioritization (highest threat processed first).
• **Trees** – Binary Space Partitioning (BSP) tree or Quadtree for spatial partitioning to optimize range queries.
More may be added as we learn further implementations.

3) Main Algorithms:

❖ Searching & Sorting Algorithms
• Linear Search – Find object in unsorted list by ID or specific attribute.
• Binary Search – Quickly locate objects when sorted by distance or threat level.
• Bubble/Insertion Sort – Simple methods for small datasets or nearly-sorted data.
• Merge/Quick Sort – Efficient O(n log n) sorting when many objects detected.
• Radix Sort – Optimal for sorting integer-based track IDs.

❖ Graph Algorithms
- Dijkstra's Algorithm for Threat Propagation & Path Planning

More may be added as we learn further implementations.

Applications in Radar System:

1. Optimal Interceptor Path Planning: Calculate safest routes for friendly aircraft avoiding high-threat zones
2. Threat Propagation Modeling: Simulate how threat levels spread through adjacent airspace sectors
3. Resource Allocation: Determine optimal placement of defense assets based on threat pathways

❖ Spatial Partitioning Algorithms

• Quadtree – For efficient range queries and collision detection in 2D airspace.
• Binary Space Partitioning (BSP) – Divide airspace into sectors for optimized searching.

4) Enhanced Data Flow

Receiving Data:

• **Simulation Mode**: Random contact generation with configurable parameters
• **File Input Mode**: Read contact data from CSV files for testing
• **Network Mode**: UDP packet reception simulating real sensor data
• **User Input Mode**: Manual contact entry for demonstration

Processing Pipeline:

1. **Data Acquisition Layer**: buffers incoming sensor data
2. **Filtering & Validation**: Validate coordinates, filter by radar range
3. **Spatial Indexing**: Build Quadtree for efficient proximity searches
4. **Classification Engine**:
    - IFF database lookup (unordered_map)
    - Threat assessment using Dijkstra-based propagation
    - Priority calculation (priority_queue)
5. **Track Management**:
    - Maintain active tracks (vector + spatial indexing)
    - Update positions using kinematic formulas
    - Handle track initiation and termination

5) Algorithm Complexity Justification:

- **Dijkstra**: O(E log V) for optimal path planning in threat networks
- **Quadtree**: O(log n) for range queries vs O(n) linear search

- **Hash Table**: O(1) IFF verification enabling real-time response
- **Priority Queue**: O(log n) threat prioritization updates

6) Conclusion:

The project will implement key concepts from the semester, making full use of Data Structures & Algorithms and Object-Oriented Programming. The initial plans are still subject to change as we learn more about the course content and delve deeper into the project. This outline is not final and will likely be revised, but it will of course revolve around this core concept.

Group Member	Registration Number	Name
Member 1	2024358	Muhammad Awab Abdullah
Member 2	2024481	Muhammad Umar Aslam
Member 3	2024027	Abdul Rehman Asif