Reflection & Justification Document

# Design Decisions & Architecture

* The simulation is structured with clear object-oriented components:  
  - Way point: encapsulates 3D coordinates of ‘n’ waypoints. The number of waypoints is decided as per the number of user inputs.  
  - Flight Path: manages path segments, time allocation, and interpolation.  
  - Functional separation for position calculation, conflict detection, drone generation, and animation ensures modularity and reusability.
* The primary drone is user-defined.
* The other drones follow parallel paths within the airspace of the primary drone by making a bounding box from the waypoints. These drones start at a random time of +-5s form the start time of the primary drone. The simulated drones are made to follow a parallel path so that there are no crashes between the simulated drones and only the first crash with the primary drone is considered.

# Path Generation through waypoints:

The current code just lets the primary drone switch direction immediately after it reaches the waypoint which is not a real scenario. We can modify the path generation such that it follows a straight line for a certain part of the segment between wi and wi+1 and then follows a circle of radius r when it crosses the first half plane and switches back to a straight line in the direction of wi+1 to wi+2. In this way we can ensure smooth trajectories.

# Spatial & Temporal Checks

* Spatial check: Uses 3D Euclidean distance between drone positions (primary vs all others) to notify if 2 drones are within certain range of each other
* Temporal check: Synchronizes all drones across uniform time steps (e.g., every second), checking each pair for proximity within a buffer radius.  
    
  Conflict is detected when both spatial and temporal proximity criteria are met.

In essence we know the trajectory of the primary drone and those of the others, if at any time t, the two drones are separated by a distance lesser than the critical distance, we consider the situation as a conflict.

# AI Integration

Currently, there is no AI integration. However, the framework is designed such that future AI modules (e.g., reinforcement learning for dynamic rerouting or neural networks for conflict prediction) could be plugged into the position or conflict-detection layers.

# Testing Strategy

1. Vary the number of simulated drones and their paths.
2. Vary the critical distance for crash to see at what boundary value of distance do the drones just escape the crash ( we can then compare this data to real world drone sizes to see how much this minimum critical distance should be for certain drones and thus we can estimate a maximum number of drones in a given airspace and given trajectories)
3. We can vary the number and location of waypoints to see whether there is a trajectory that helps us avoid a conflict given the number and trajectories of the simulated drones.

# Scalability for Real-World Data

To scale for tens of thousands of drones, key improvements would be needed:  
1. Vectorized computation (e.g., via NumPy or GPU acceleration) for position and distance calculations.  
2. Spatial indexing (e.g., k-d trees or bounding volume hierarchies) to avoid O(n²) comparisons.  
3. Time slicing with interval trees to skip irrelevant drones per timestep.