UAV Deconfliction System: Design Reflection

1. Architectural Choices & Design Decisions
The UAV Deconfliction System is designed using a clean, three-layer modular architecture to keep components independent and maintainable:

Presentation Layer – Streamlit UI
Choice: Streamlit was chosen for its ease of use and ability to quickly prototype interactive interfaces.

Why it works: It allows fast development without needing deep frontend expertise, while still offering powerful visualization through Plotly, especially for 3D route displays.

Business Logic Layer – Path Calculation Engine
Design: Built using stateless, pure functions that handle tasks like collision detection and path interpolation.

Reasoning: This keeps the logic predictable, testable, and independent of the user interface. All mathematical computations are encapsulated, making the core logic reusable and modular.

Data Layer – Session State
Implementation: Streamlit’s session state is used to preserve user input and system state across interactions.

Key Design Decisions:

3D Euclidean distance is used for spatial conflict detection — it’s simpler and sufficient for small-scale airspace.

Linear path interpolation provides a balance between accuracy and performance when estimating positions between waypoints.

Time-window comparison using HHMM format makes sorting and conflict detection straightforward and human-readable.

2. Spatial & Temporal Conflict Detection

Spatial Conflict Detection

python

```
def detect_collisions(mission1, mission2, safety_margin=1.5):
    path1 = interpolate_path(mission1)
    path2 = interpolate_path(mission2)
    return [p1 for p1 in path1 for p2 in path2
            if euclidean_distance(p1, p2) < safety_margin]
```

How it works:

Each mission path is interpolated into 10 evenly spaced points per segment (customizable).

Every point in one path is compared with every point in the other to find potential collisions.

Safety Buffer: A 1.5-meter margin is added to account for drone size and GPS drift.

Temporal Conflict Detection

Method: Start and end times are compared to check for overlapping missions.

Current Assumption: Drones move at constant speed between waypoints.

Planned Upgrade: Future versions will support variable velocity profiles, enabling more accurate time-position analysis.

3. Testing Strategy & Edge Case Handling

Edge Cases Addressed

<u>*Simultaneous Waypoint Arrival:*</u>

When drones reach the same coordinate at the same time, a conflict is detected.

Fix: A minimum time buffer is enforced between overlapping missions.

<u>*Vertical Stacking:*</u>

When drones occupy the same X-Y position but different altitudes (Z-axis), they are currently treated as non-conflicting.

This keeps airspace usage flexible while preventing false alarms.

<u>*Single-Waypoint Missions:*</u>

To allow proper collision detection, a 1-meter hover radius is automatically assumed around single-point missions.

<u>*Validation Strategy*</u>

Automated Testing: Core algorithms are tested using pytest to ensure reliability and catch edge cases early.

4. Scaling to 10,000+ Drones

To prepare the system for real-world scenarios involving thousands of drones, scalability was built into the design:

Optimization Techniques

<u>Spatial Partitioning:</u>

Airspace is divided into 100m³ voxel grids.

Collision checks are only performed for drones in the same or neighboring voxels.

Approximate Conflict Checks:

A broad-phase filtering step uses bounding boxes to rule out obviously safe paths.

Only ~5% of pairs go through fine-phase precision checks.

Parallel Processing:

Distance calculations are offloaded to the GPU when available.

For large datasets, Apache Spark can be used to distribute processing across multiple nodes efficiently.

CONCLUSION

The UAV Deconfliction System demonstrates a robust, modular approach to managing shared airspace, balancing clarity, performance, and scalability. Through thoughtful architectural decisions—like separating the user interface from the business logic and leveraging pure functions for computation—the system remains easy to maintain and extend.

Our current implementation effectively handles both spatial and temporal conflict detection, with safeguards in place for edge cases that commonly arise in real-world operations. While it already supports small-to-medium scale simulations with reliable accuracy, planned enhancements such as dynamic velocity profiles and GPU-accelerated processing will further improve precision and scalability.

Looking ahead, this system lays a strong foundation for real-time drone traffic management, opening the door to safe, autonomous operation in increasingly crowded skies.