

# Distributing a Fleet of Drones over an Area with No-Fly Zones

**sdmay25-21**

**Members:** Nicholas Kokott, Melani Hodge, Cole Stuedeman, Everett Duffy, Ken Schueman,  
Samuel Russett

**Client/Advisor:** Professor Goce Trajcevski



# Project Overview

- Goals:
  - To be able to give a UI to users that displays their drones interacting with their points of interest for whatever reason they have given.
  - Have drones fly in a shortest path to certain events while ensuring that they get around no-fly zones.
- Importance:
  - Many drone users currently have to manually use them to respond to events, we will automate this so that drones can instantly perform the jobs needed.
  - Rather than users controlling drones one by one they can now have all of them move at the same time assuming there are multiple events happening simultaneously.






UI Design

## Window

SkySector

Home  
Favorites  
Map View



+ Add new


## Window

SkySector

Home  
Favorites  
Map View

Please add in your input values for the map, no-fly zones, and number of drones.

Map Longitude  Map Latitude

No-Fly Zone  

Drones

+ Partition Map

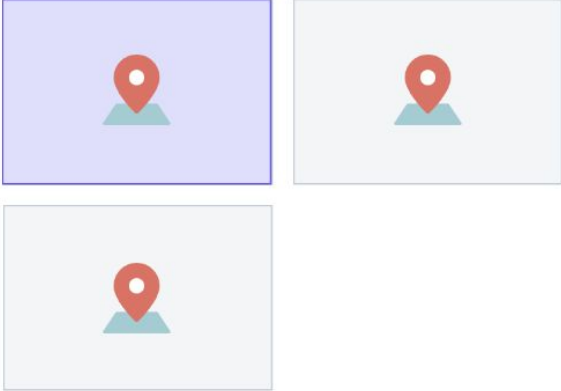
+ Add new

## Window

SkySector

Home  
Favorites  
Map View

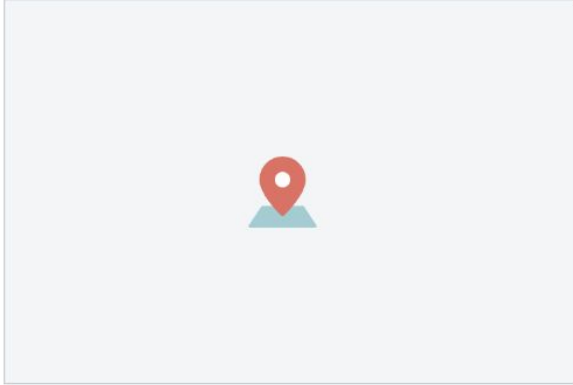
+ Add new






## Window

SkySector

Home  
Favorites  
Map View



Drone Information		Partition Information
Drone 1		Partition 1
Drone 2		Partition 2
Drone 3		Partition 3

+ Add Drone

## Configuration

Number of Drones:

Algorithm Selection:

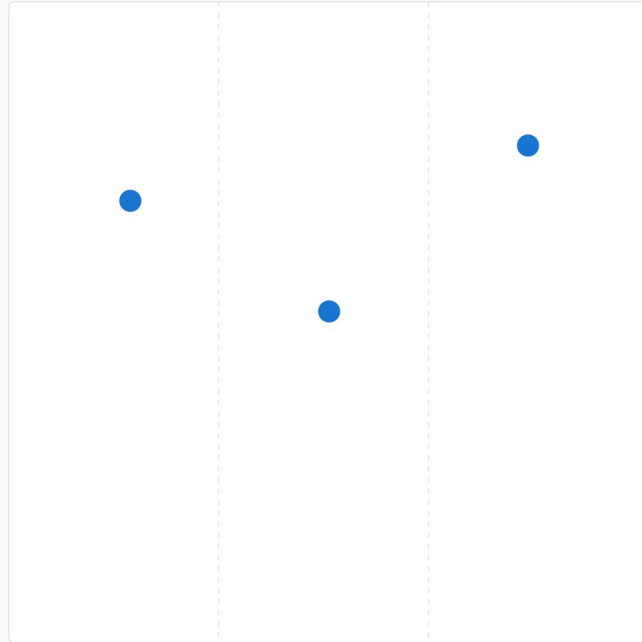
No-fly Zone Input

## Simulation Controls

Start

Stop

## Event Management

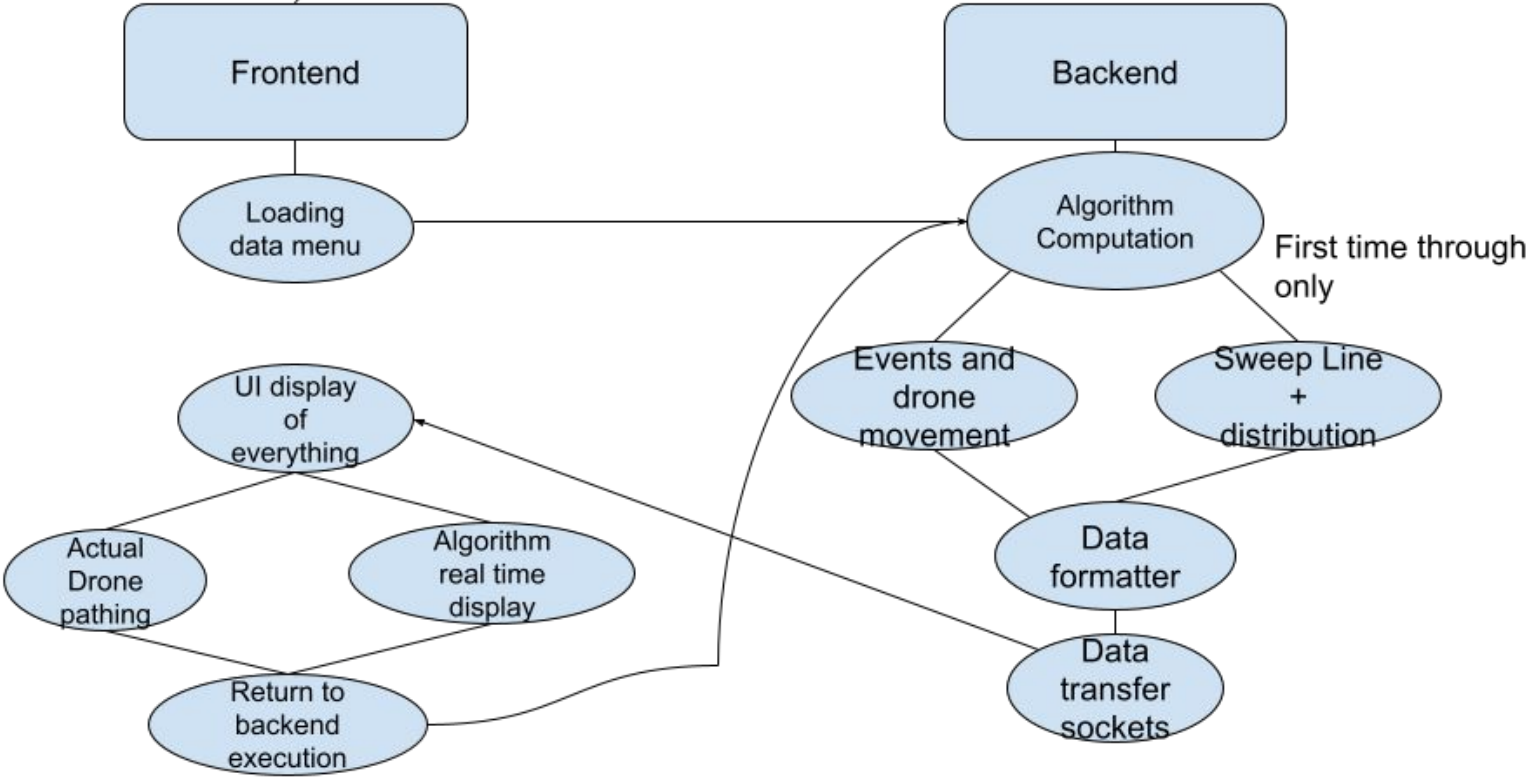


● Drone ■ No-fly Zone - - - Partition Boundary



# Global Architecture

# Website Architecture

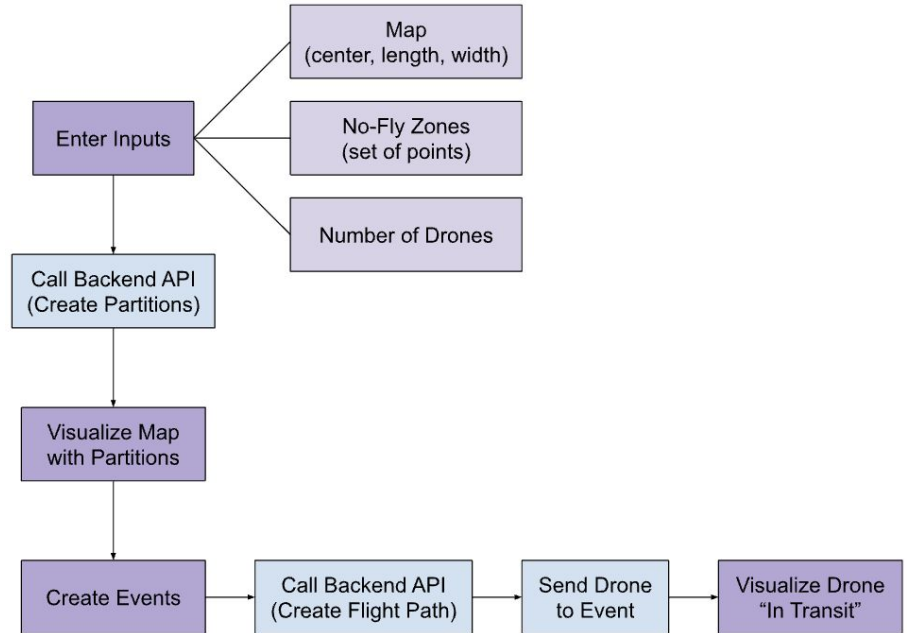




# Functionality



- User Interaction (Purple)
  - Enter inputs
  - Select events within partitions
- Backend (Blue)
  - Calls APIs
  - Performs algorithmic functions to:
    - Create partitions
    - Create flight path





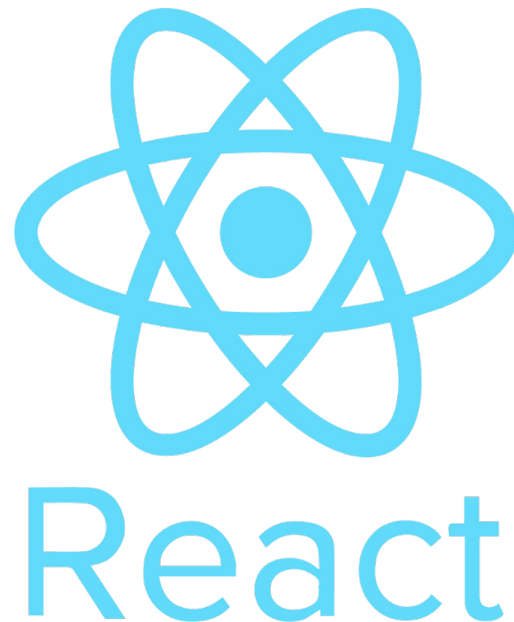
# Technology Considerations

# Frontend



## React (Redux Toolkit)

- **Strengths:**
  - Offers component-based architecture
  - Easy to find tools and libraries for common features (large ecosystem)
- **Weaknesses:**
  - Complex management in large applications
  - Requires extra tools like Redux for state management
- **Trade-offs:**
  - Flexible BUT requires extra decisions on structuring code and handling state
- **Alternatives:**
  - Vue (easier learning curve, great for reactive applications)
  - Svelte (more performant but has smaller ecosystem)



# Frontend



## Typescript

- **Strengths:**
  - Adds static typing to JavaScript
    - Reduces run-time errors
    - Improves code readability
    - Allows for better IDE support
- **Weaknesses:**
  - Requires a learning curve
  - Compilation can slow down development if not optimized
- **Trade-offs:**
  - Type safety at cost of additional setup and debugging time
- **Alternatives:**
  - JavaScript (simplistic but lacks robustness and development efficiency)



**IS IT WORTH THE TYPE?**

# Frontend



## Mapbox API

- **Strengths:**
  - Highly performant for rendering large map data
  - Offers extensive customization and interactivity
- **Weaknesses:**
  - Can be costly at scale
  - Learning curve for complex interactions
- **Trade-offs:**
  - Powerful BUT requires expertise in geospatial data handling
- **Alternatives:**
  - Leaflet (lighter and open-source but offers less features)
  - Google Maps API (more familiar but limited in customization)



# Backend



## Django (Python)

- **Strengths:**
  - Includes a comprehensive set of tools, reduces need of third-party dependencies
  - Secure
  - Works well with PostgreSQL and PostGIS
- **Weaknesses:**
  - May lead to inefficiencies in more modular, microservices-based design
  - Due to the API-based application, need to use Django Rest Framework (DRF) as well
- **Trade-offs:**
  - Rapid development with security BUT may add overhead with complex API-only design
- **Alternatives:**
  - Flask (lighter but requires more configuration for security)
  - FastAPI (highly performant and asynchronous, great for real-time data)



# Backend



## PostgreSQL with PostGIS

- **Strengths:**
  - PostgreSQL (robustness for data management)
  - PostGIS (spatial data processing)
  - Ideal for complex geospatial data queries
- **Weaknesses:**
  - Can be demanding on resources for high query loads
  - Requires database management expertise
- **Trade-offs:**
  - Highly capable of complex spatial operations BUT resource intensive
- **Alternatives:**
  - MySQL with GIS extensions (limited geospatial functions)
  - MongoDB with geospatial indexing (simpler, document-based data but lacks advanced operations)

PostgreSQL



PostGIS



# Areas of Concern and Development



- **Concern:** Real-time geospatial calculations for multiple drones could be resource intensive.
- **Solution:** Consider implementing caching for frequently accessed data. Also consider implementing a background processing tool to offload heavy computations from main API.
  
- **Concern:** If algorithms provided are intensive, they may cause latency issues.
- **Solution:** Preprocess some data or use a worker system to reduce delays in partitioning calculations.
  
- **Concern:** The interface must remain responsive and user-friendly, even with high data loads or complex visuals.
- **Solution:** Ensure efficient state management in React and use lazy loading in Mapbox to avoid frontend slowdowns.





# Conclusions

In conclusion, the drone fleet management project aims to optimize drone response time for various user scenarios, including search and rescue, delivery, and infrastructure maintenance, while accounting for no-fly zones and other obstacles. By having utilizing Agile alongside our task decomposition and milestones we will be able to develop this project as well as anyone could with great speed and accuracy. Mitigating and understanding our potential risks will also allow us to develop with security in mind ahead of time, which will make it much easier to keep our users resources safe.