# Distributing a Fleet of Drones Over a Region with No-Fly Zones

**Team:** sdmay25-21

**Client/Advisor:** Professor Goce Trajcevski

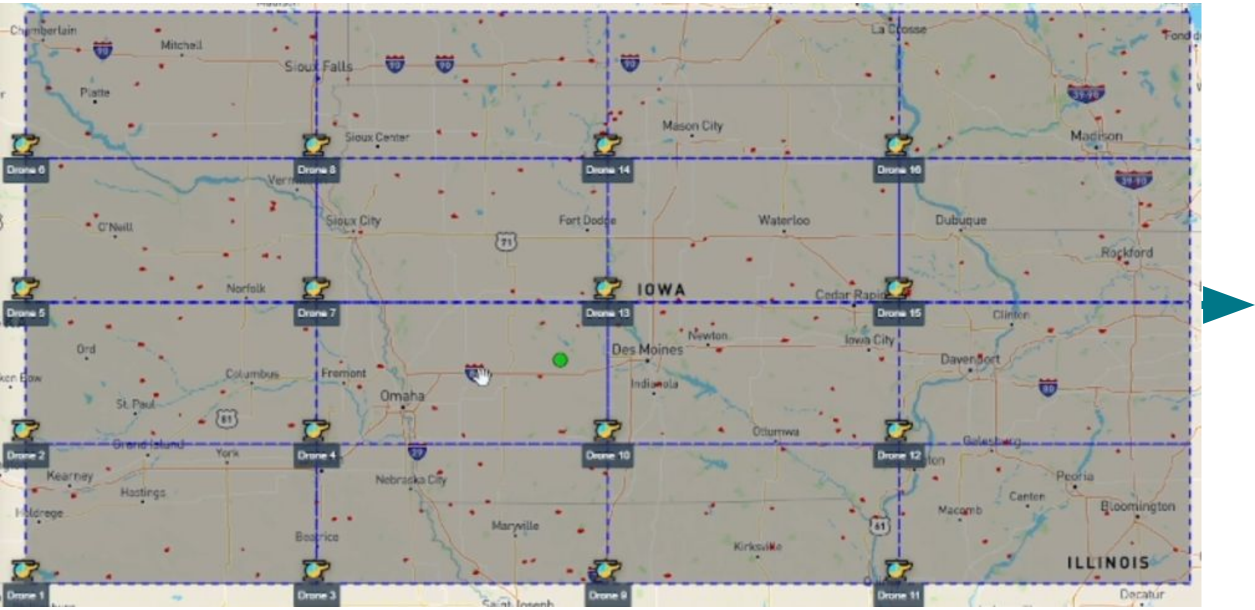
https://sdmay25-21.sd.ece.iastate.edu/

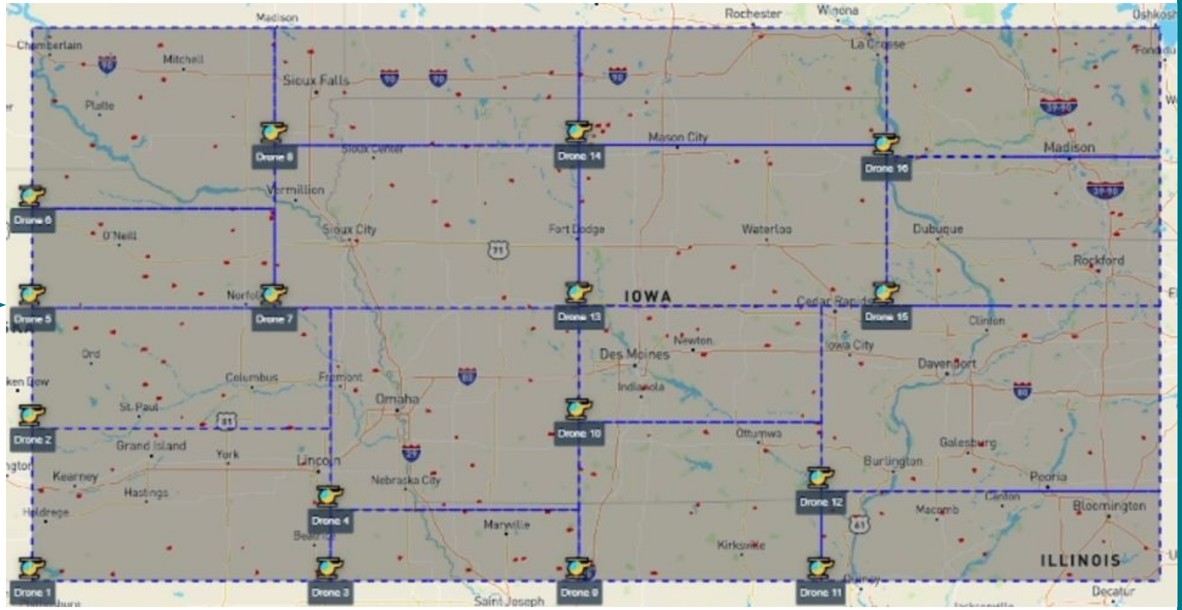
# Team

- Nicholas Kokott - Team Organizer
- Melani Hodge - Frontend Design/Testing
- Everett Duffy - Component/Module Design
- Cole Stuedeman - Testing
- Kenneth Schueman - Advisor Communication
- Samuel Russett - Research Discovery and Testing

# Project Overview

- Goals:
  - Minimize worst case response time through partitioning
  - Shortest Path/No Fly Zone Avoidance
- Objective:
  - Provide UI Displaying Drone Flight Interaction
  - Automatic event management





# Target Users

- Delivery

  - Package delivery in crowded urban areas

- Search and Rescue

  - Emergency response and location

- Civilian Hobby Flight

  - Avoidance of military airspace and other restricted areas

# Requirements

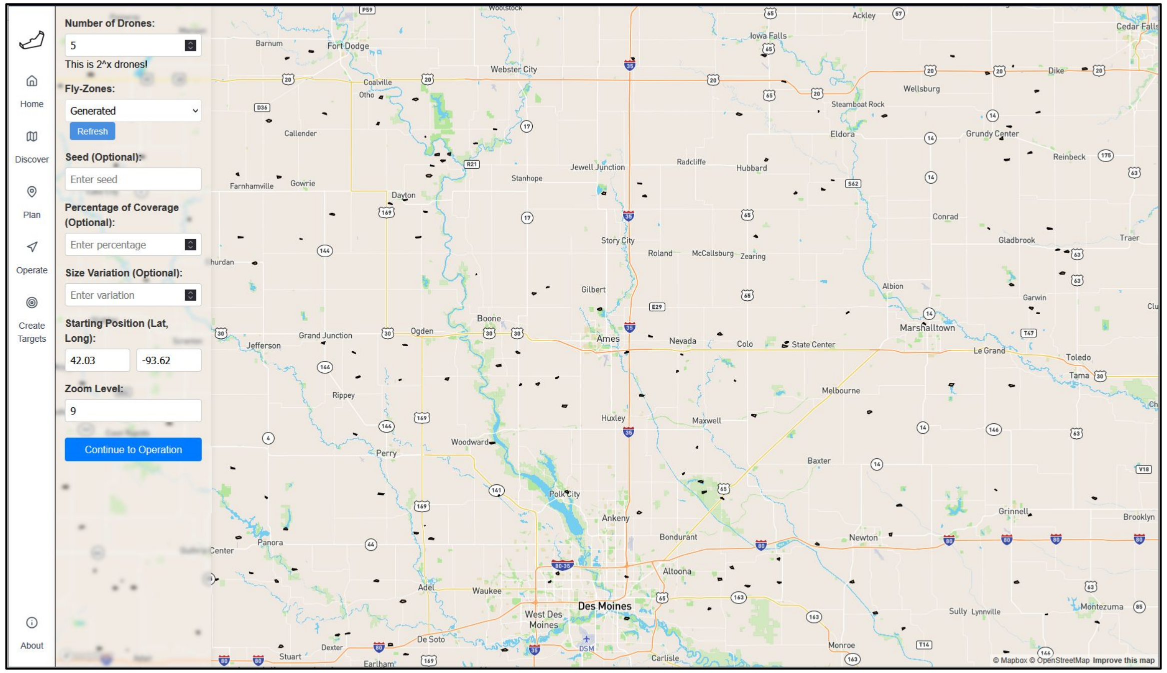
# Non-Functional Requirements and Constraints

- Flexibility in configuration
- Architecture provided by ETG
- Consider response limitations
- Consider cost constraints

**The full list of non-functional requirements can be found in our design document (p.8)**

# Functional Requirements

- Enable algorithm selection
- Allow users to…
  - input number of drones
  - input events
  - start and top simulation
  - navigate UI
- Ease of use for those unfamiliar with drone use or code.



Number of Drones: 5
This is 2^x drones!

Fly-Zones: Generated

Seed (Optional): Enter seed

Percentage of Coverage (Optional): Enter percentage

Size Variation (Optional): Enter variation

Starting Position (Lat, Long): 42.03 -93.62

Zoom Level: 9

**The full list of functional requirements can be found in our design document(p. 7-8)**

# Standards

- IEEE 1471 (Software Architecture Standard)
    - Needs real-time interactions
    - Multiple components(UI, algorithms, eternal API)
    - Architecture documentation

- ISO/IEC 27000 (Information Security)
    - The requirements explicitly mention security concerns about attackers
    - Each session needs to be unique and secure
    - The system handles real-world location data that must be protected

# System Design - Global Architecture

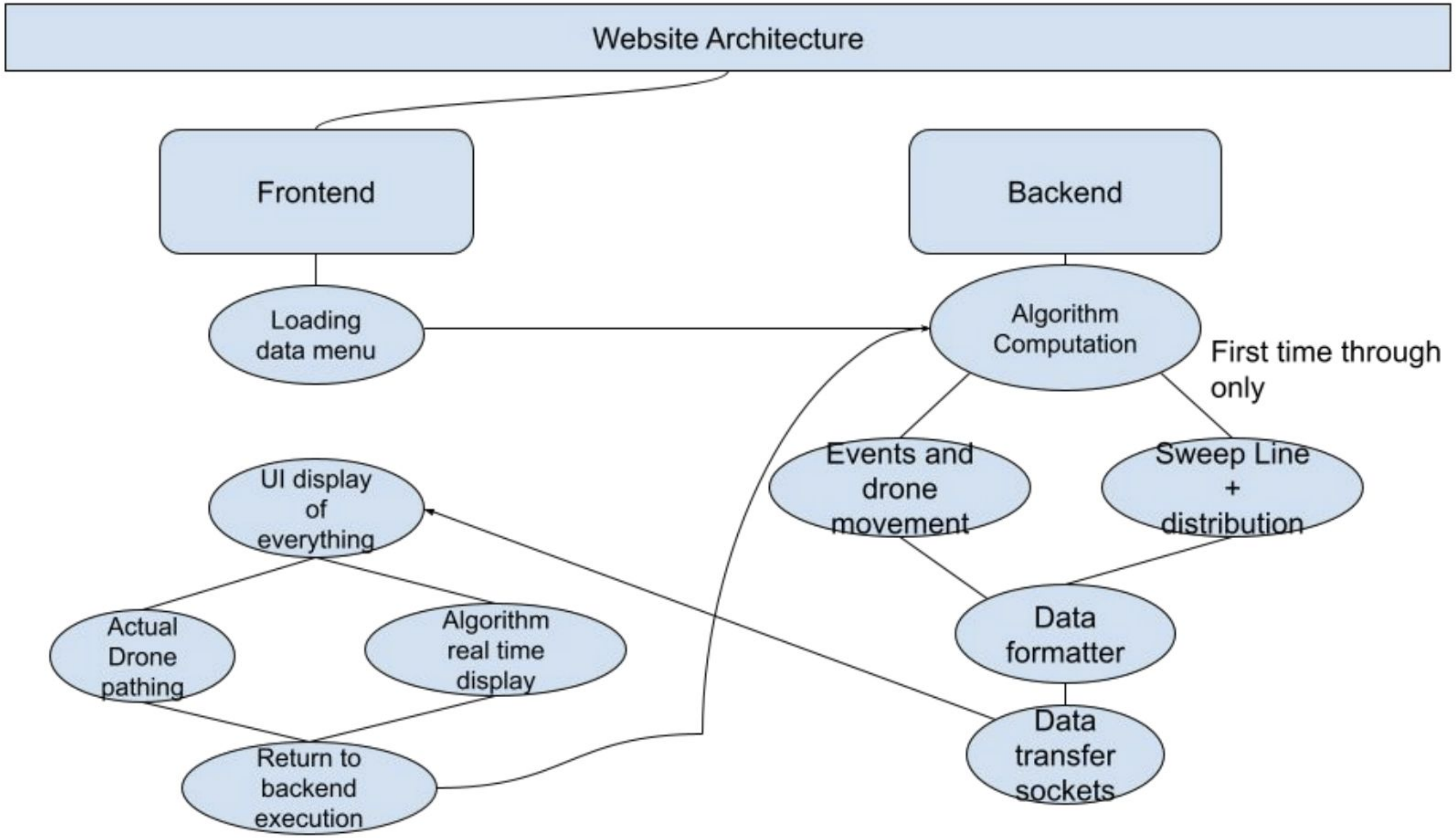
# Global Architecture

- Take data from frontend and push to backend

- Do backend calculations (partitioning) and display back to frontend

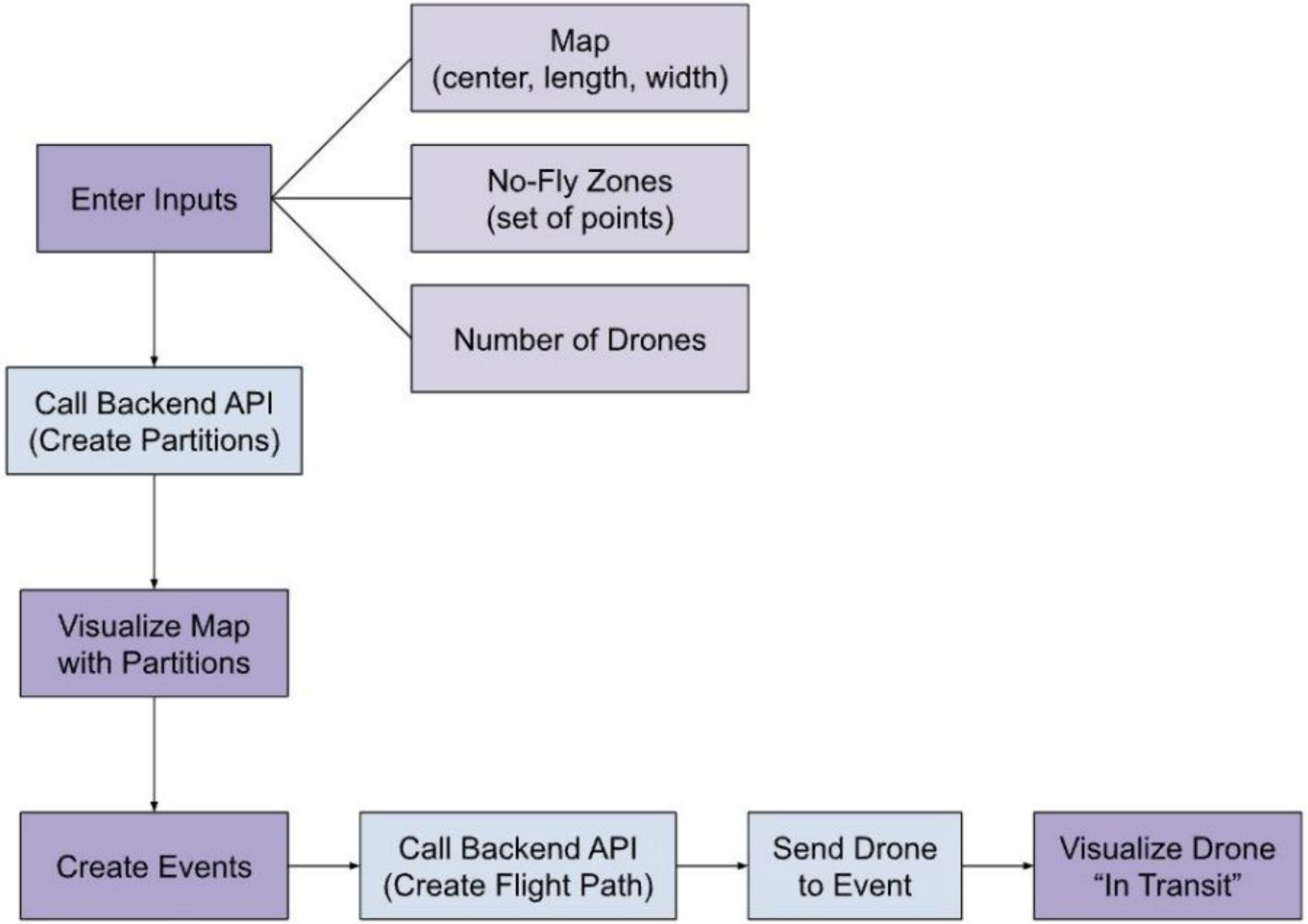
- Have users input events, and push locations to backend to handle

- Relay back to frontend and show drone pathing



# Functionality

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



# System Design - Frontend

# Frontend Foundation

## React (Vite Toolkit)

- **Strengths**: Component-based, large ecosystem for tools, support for TypeScript
- **Weaknesses**: Complex in large apps, needs extra tools for state management
- **Trade-offs**: Flexible **but** requires structuring and state decisions
- **Alternatives**: Vue (easier learning), Svelte (faster, smaller ecosystem)

https://iconduck.com/icons/13180/react-original-wordmark

https://iconduck.com/icons/13281/vite-watermark

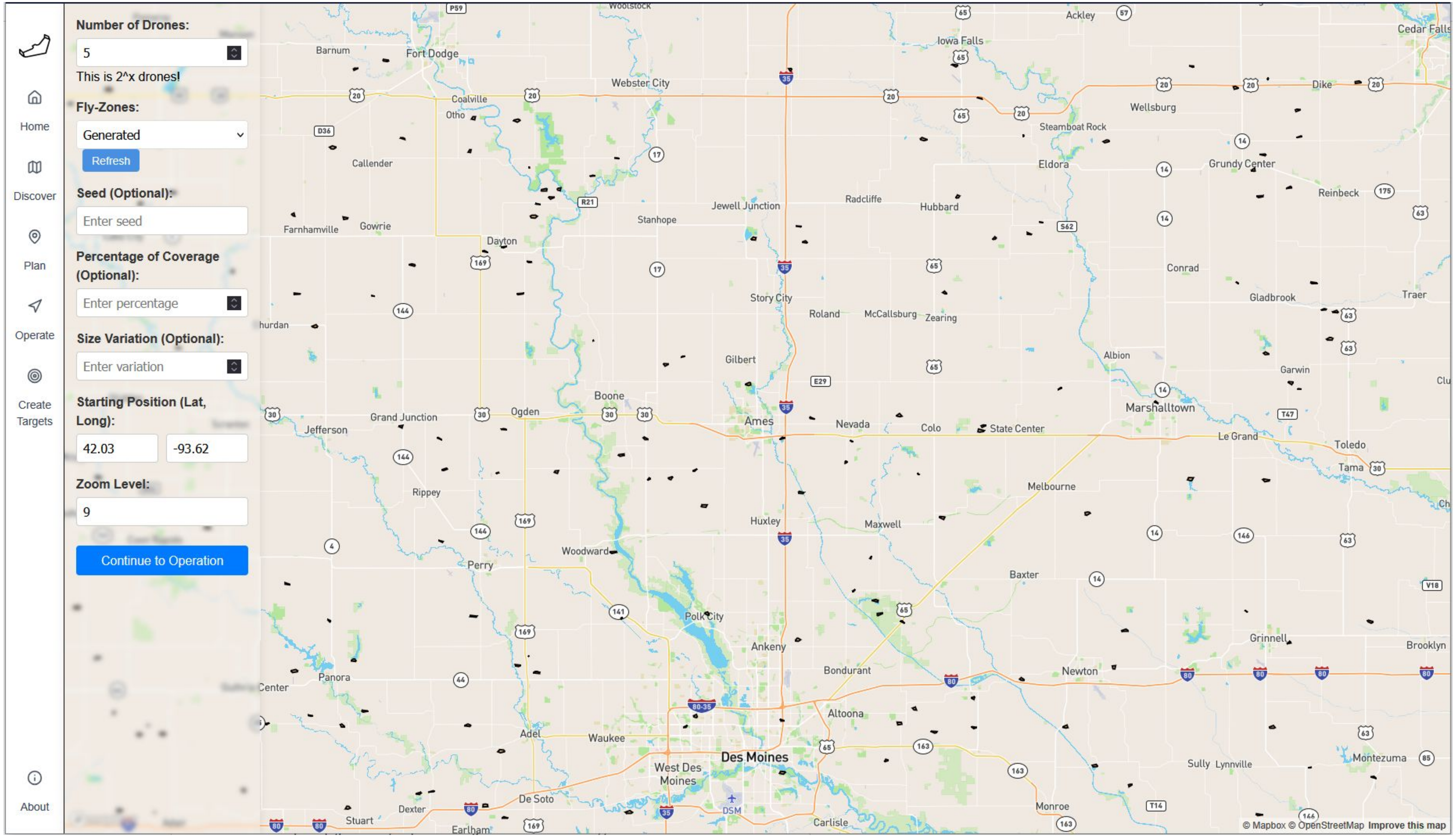
# Frontend Specifics

**MapBox**

- **Strengths**: Open Source, endlessly customizable, and free
- **Weaknesses**: Complex to render objects, shapes, and text elements
- **Trade-offs**: As stated above its Open Source and customizable, so we had to create our own features
- **Alternatives**: Google Maps, GeoServer, and MapServer

https://iconduck.com/icons/17340/rmapbox

# UI Design - No Fly-Zone selection and partitioning



**Number of Drones:** 5

This is 2^x drones!

**Fly-Zones:** Generated

**Seed (Optional):** Enter seed

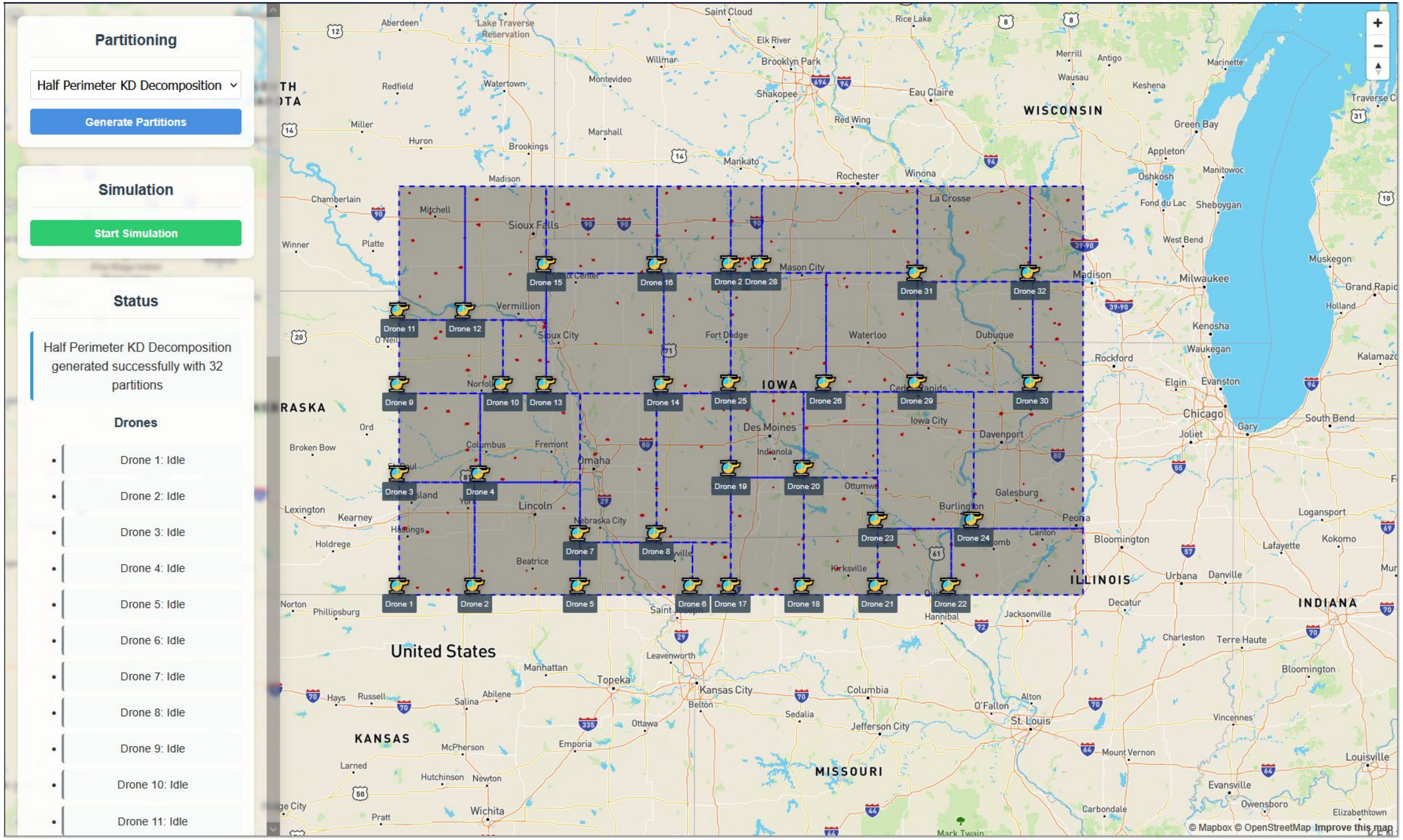
**Percentage of Coverage (Optional):** Enter percentage

**Size Variation (Optional):** Enter variation

**Starting Position (Lat, Long):** 42.03 -93.62

**Zoom Level:** 9

# UI Design - No Fly-Zone selection and partitioning



Partitioning

Half Perimeter KD Decomposition ⌄

Generate Partitions

Simulation

Start Simulation

Status

Half Perimeter KD Decomposition generated successfully with 32 partitions

Drones

- Drone 1: Idle
- Drone 2: Idle
- Drone 3: Idle
- Drone 4: Idle
- Drone 5: Idle
- Drone 6: Idle
- Drone 7: Idle
- Drone 8: Idle
- Drone 9: Idle
- Drone 10: Idle
- Drone 11: Idle

# System Design - Backend

# Backend

**Django (Python)**

- **Strengths**: Comprehensive tools, secure, great with PostgreSQL/PostGIS
- **Weaknesses**: Overhead in microservices and API setups
- **Trade-offs**: Fast, secure **but** less modular
- **Alternatives**: Flask (lighter), FastAPI (async, real-time)

https://en.wikipedia.org/wiki/Python_%28programming_language%29

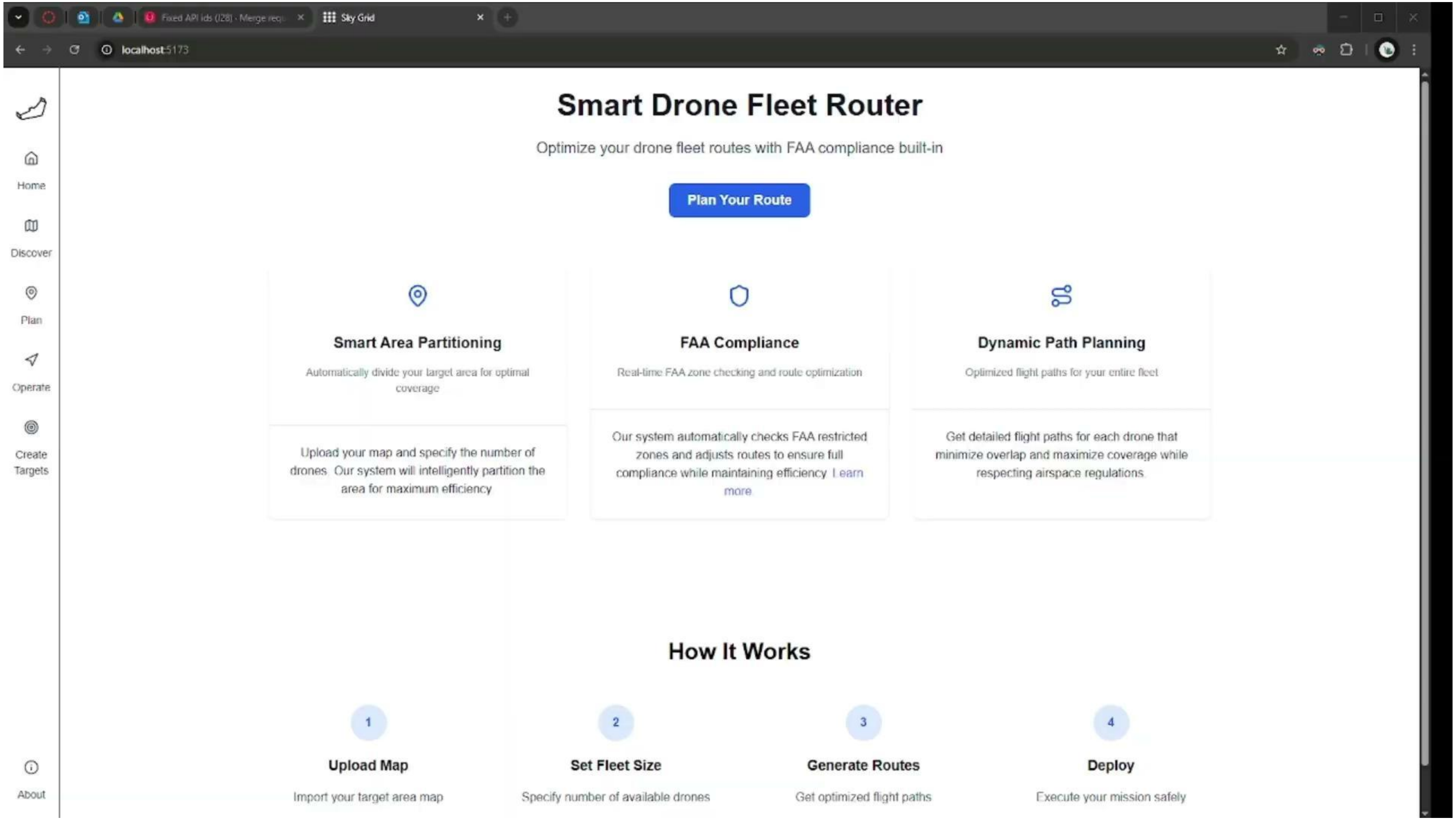
# Backend

**PostgreSQL with PostGIS**

- **Strengths**: Robust, excels in complex geospatial queries
- **Weaknesses**: Resource-heavy, requires expertise
- **Trade-offs**: Powerful **but** demanding
- **Alternatives**: MySQL (limited GIS), MongoDB (simpler, fewer features)

https://www.linkedin.com/pulse/aprender-postgresql-y-postgis-juan-carlos-jaramillo-ceballos-t4uhe/

# Demo Video



## Smart Drone Fleet Router

Optimize your drone fleet routes with FAA compliance built-in

### Smart Area Partitioning

Automatically divide your target area for optimal coverage

Upload your map and specify the number of drones. Our system will intelligently partition the area for maximum efficiency

### FAA Compliance

Real-time FAA zone checking and route optimization

Our system automatically checks FAA restricted zones and adjusts routes to ensure full compliance while maintaining efficiency. Learn more

### Dynamic Path Planning

Optimized flight paths for your entire fleet

Get detailed flight paths for each drone that minimize overlap and maximize coverage while respecting airspace regulations

## How It Works

1. **Upload Map**  
Import your target area map

2. **Set Fleet Size**  
Specify number of available drones

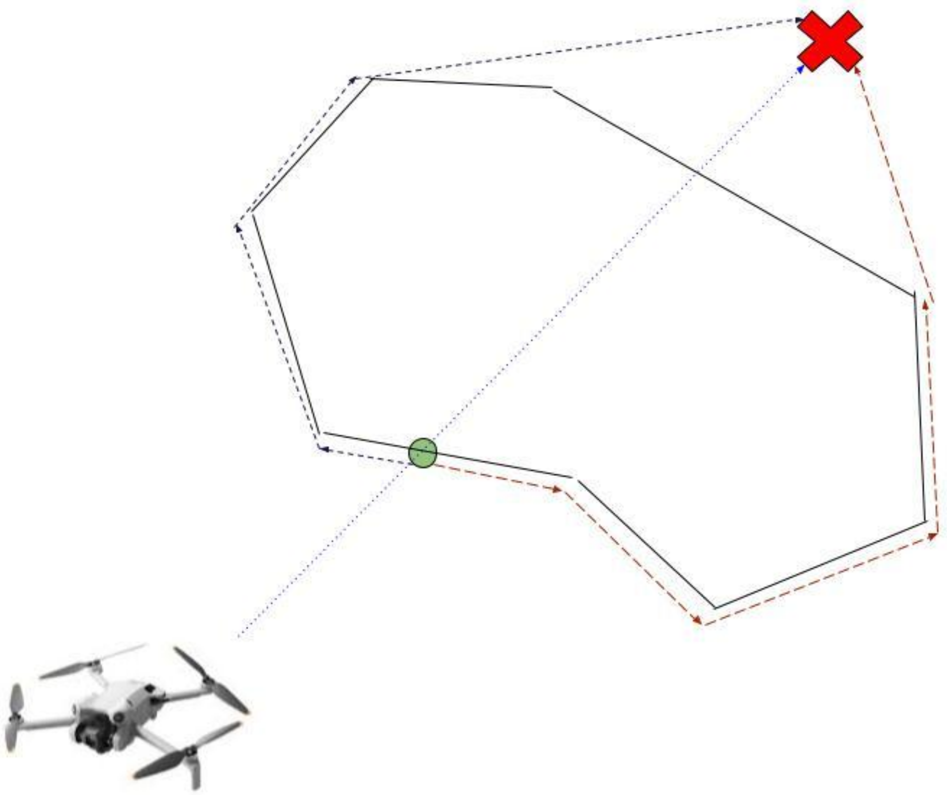
3. **Generate Routes**  
Get optimized flight paths

4. **Deploy**  
Execute your mission safely

# Challenges

# Main Issues Faced

- Pathfinding the Drones
  - The way the database was setup did not allow for the drones to pathfind in a traditional algorithm (Dijkstras, AStar, etc.)
  - Instead, we precomputed the path while checking if no fly zones were there
- Integration with Grad Student's code
  - No-fly zone generation and partitioning algorithms were given, but documentation was poor
  - To solve this problem we had to trace through the code manually to determine what needed to be passed through



# Risks & Mitigation

- Performance Issues
  - Real time performance, Algorithm scalability, Browser compatibility
- User Data Security
  - Injection vulnerabilities, data accuracy, input validation
- Testing All Use Cases
  - Testing tools

# Testing

# Testing

- Unit Tests
  - Test individual components and functions in isolation (frontend, backend, and integration logic)
  - Use Vitest for frontend tests (entering drone/user info in React forms)
  - Backend unit tests validate model behavior and data flow with GSA and Mapbox compatibility

- Interface Tests
  - Test communication between Backend, Frontend, GSA, and Mapbox
  - Validate API endpoints (NoFlyDataViewTests) and request/response structure
  - Use serializer tests (NoFlySerializerTests) to confirm data transformation accuracy
  - Mock external APIs to test parameter handling and error/success responses

# Testing

- System Testing
  - Test complete workflows and system functionality end-to-end
  - Run drone path planning tests with no-fly zones (DronePathNoFlyTest)
  - Validate integration of components like frontend, backend, GSA, and FAA data loaders

- Regression Tests
  - Ensure new changes don’t break existing features
  - Use reusable test suites and specific regression cases (ModelsRegressionTests)
  - Validate data integrity and behavior across edge cases

# Conclusion

# Final Thoughts

- Satisfied user requirements with efficient development techniques

- Be quick on your feet when it comes to necessary changes

- Future work and enhancements:
  - Custom number of drones
  - Drone/Application connection
  - Expand country geojson

https://freedesignfile.com/767114-unmanned-aerial-vehicle-drone-clipart/#google_vignette

# Questions?

# Testing

- Acceptance Testing
  - Verify system meets all functional and non-functional requirements
  - Use case-based testing focused on real-world user workflows
  - Ensure critical features behave correctly from the user's perspective

# Project Plan

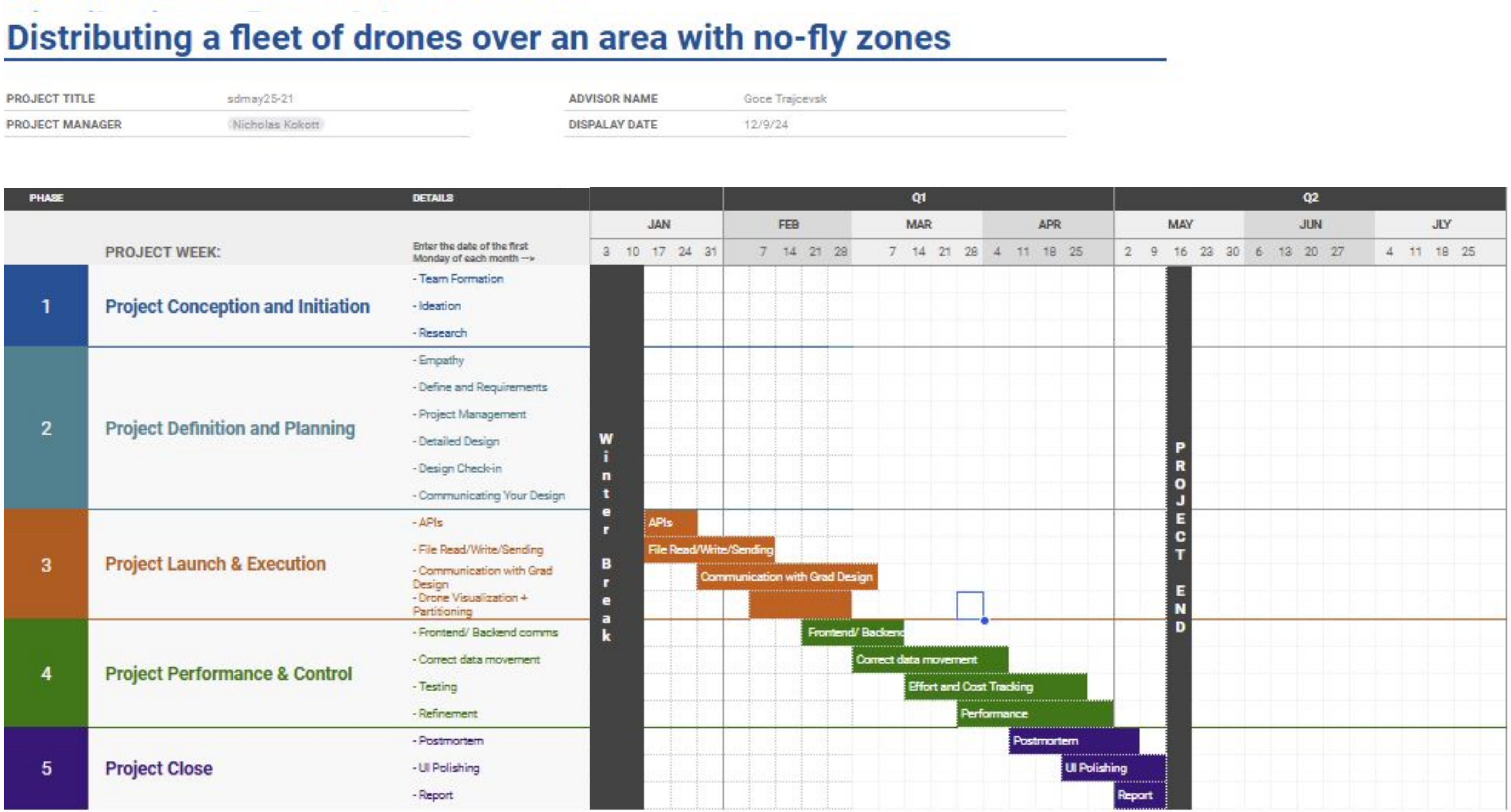
# Project Plan - Tasks

- Set up frontend (React+Vite) and Backend (Python + PostgreSQL)
- Develop Communication sockets
- Algorithm implementation
- Incorporate MapBox API
- Develop UI and backend API
- Test everything
- Develop input systems

# Project Plan - Risks & Mitigation

- Performance Issues
  - Real time performance, Algorithm scalability, Browser compatibility
- User Data Security
  - Injection vulnerabilities, data accuracy, input validation
- Testing All Use Cases
  - Testing tools

# Project Plan - Gantt Chart (Spring 2025)



## Distributing a fleet of drones over an area with no-fly zones

| | | | |
|---|---|---|---|
| PROJECT TITLE | sdmay25-21 | ADVISOR NAME | Goce Trajcevsk |
| PROJECT MANAGER | Nicholas Kokott | DISPALAY DATE | 12/9/24 |

| PHASE | | DETAILS |
|---|---|---|
| | PROJECT WEEK: | Enter the date of the first Monday of each month --> |
| 1 | Project Conception and Initiation | - Team Formation<br>- Ideation<br>- Research |
| 2 | Project Definition and Planning | - Empathy<br>- Define and Requirements<br>- Project Management<br>- Detailed Design<br>- Design Check-in<br>- Communicating Your Design |
| 3 | Project Launch & Execution | - APIs<br>- File Read/Write/Sending<br>- Communication with Grad Design<br>- Drone Visualization + Partitioning |
| 4 | Project Performance & Control | - Frontend/ Backend comms<br>- Correct data movement<br>- Testing<br>- Refinement |
| 5 | Project Close | - Postmortem<br>- UI Polishing<br>- Report |

# Ethics and Professional Responsibility (extra slides)

# Areas of Professional Responsibility

- Work Competence
    - Aim to get our work done quickly and with quality
- Communication Honesty
    - Always be honest with the team and problems we are facing
- Social Responsibility
    - Make the application with users in mind to benefit them

# Virtues:

- Collaboration
  - Work well together and be honest
- Respect
  - Treat team members well and listen to them
- Accountability
  - Hold people accountable for their work